Learn with
Google AI | Explore ML

Crowdsource
By Google

prastuti'23

Register at

https://prastuti.in/

and join the

Whatsapp Group

Learn with
Google AI | Explore ML

Crowdsource
By Google

RECOGNIZANCE WORKSHOP 2

An Introduction to Machine Learning

Learn what is Machine Learning and the different algorithms used in it.

Learn with
Google AI | Explore ML

Crowdsource
By Google

Introduction to Machine Learning

1

Introduction

What is Machine Learning and Different categories of Machine Learning Algorithms.

2

Supervised Learning

- Linear Regression
- Logistic Regression

3

Neural Networks

- Introduction
- Activation Functions
- Gradient Descent
- Backpropagation

4

Hidden Neural Networ

- Deep Networks
- Forward and Backward propagation
- Parameters vs Hyperparameters

5

Unsupervised Learning

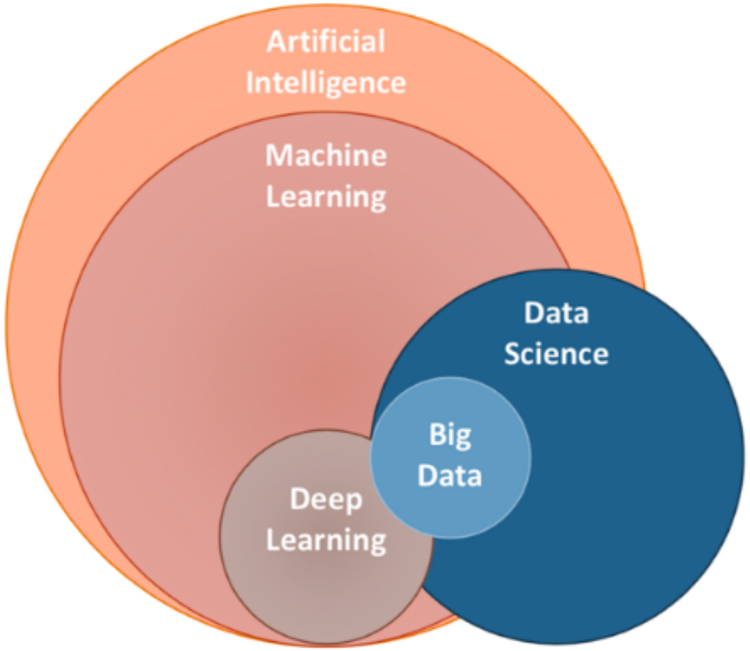
- K-means clustering
- KNN (k-Nearest neighbours)

What is Machine Learning?

- Well, according to Arthur Samuel, Machine Learning is a field of study that gives computers the ability to learn without being explicitly programmed. (1959)

- Basically, The goal is that we train the Machine in such a way that its able to perform tasks on its own. It is one of the three main branches of Artificial Intelligence, others being Computer Vision/ Deep Learning and Reinforcement Learning.

- Machine Learning basically deals with analytical data and algorithms.



Some Examples of the Tasks...

- Suppose you have to predict housing prices, given you have the prices of some random houses with varying features such as no. of rooms, area of the flat, etc. This type of problems, which have to predict continuous valued output (As price can be any number) is known as Regression Problem.

- Suppose you have to predict whether a given image is of a dog or a cat, given you have photos of some already classified images of dogs and cats. This type of problems, which have to predict discrete valued output (As the picture can be wither dog or cat) is known as Classification Problem.

Machine Learning In a Brief

The above problems are tackled by creating an ML Model. A Model is an architecture which uses up some algorithm to extract useful information from the data, e.g- the prices of the house already given. These ML algorithms can be further divided into two segments -

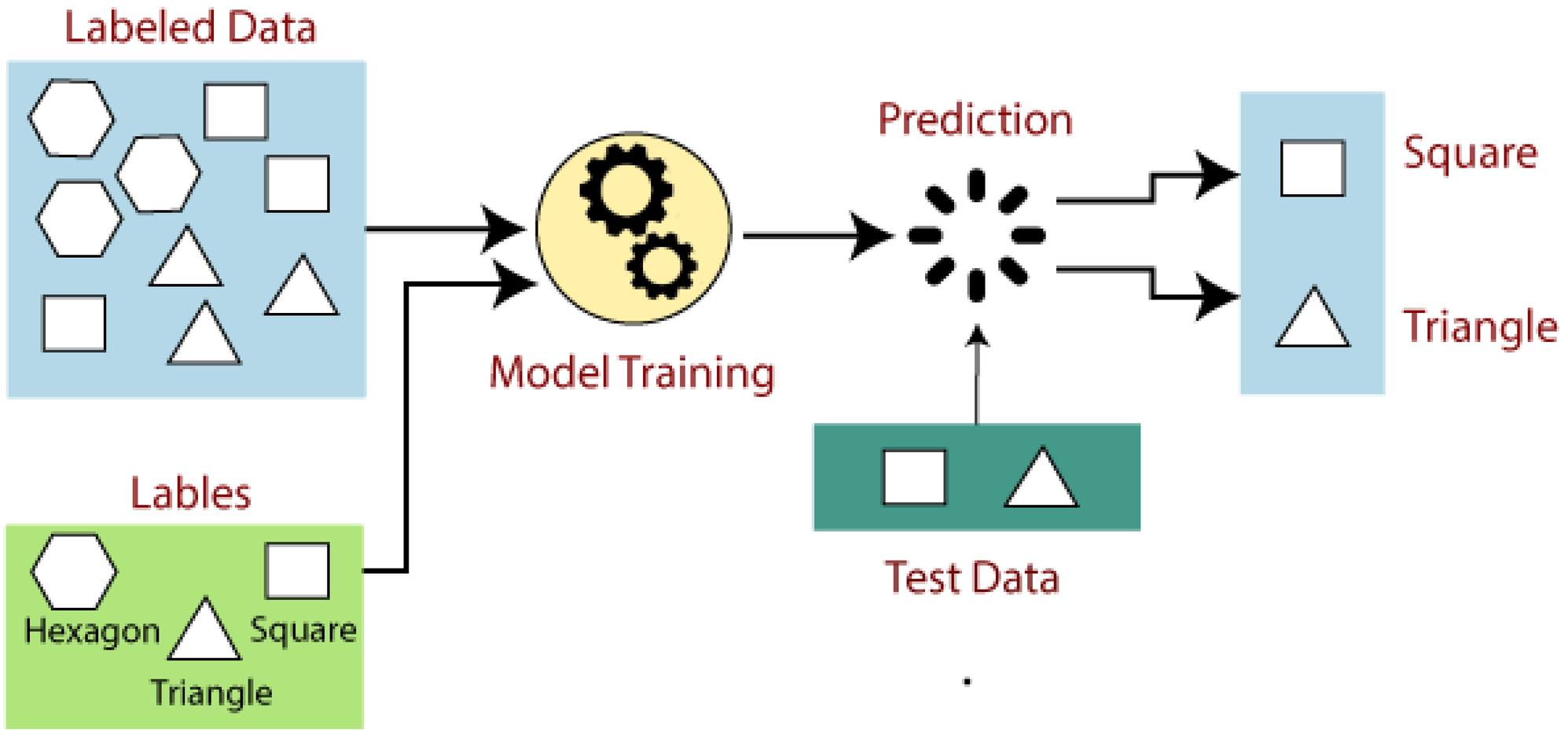
- Supervised Learning- These algorithms are designed to train the data which has been labelled for a particular output.
- Unsupervised Learning- These algorithms are designed to train the data which are neither classified nor labelled. The algorithms generally try to identify a pattern in the input data points.

Supervised Learning

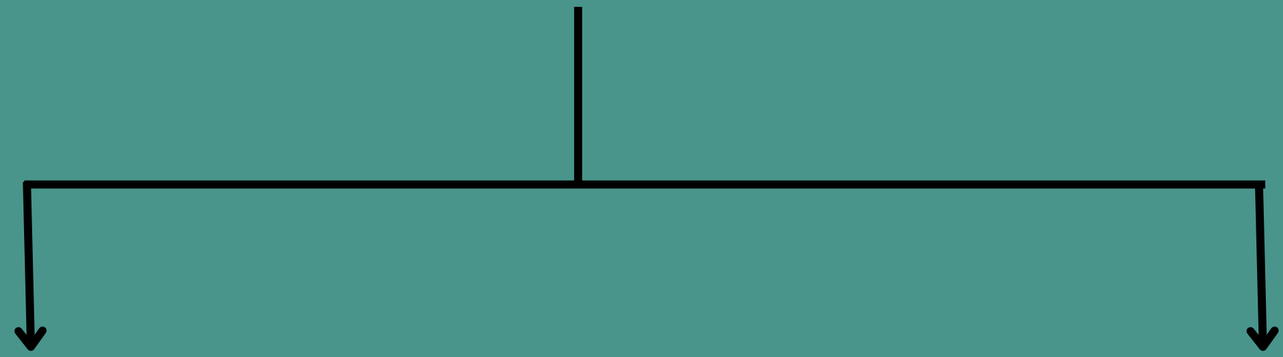
What is Supervised learning?

- In supervised learning, models are trained using labelled datasets, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.
- The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y).
- In the real-world, supervised learning can be used for Risk Assessment, Image classification, Fraud Detection, spam filtering, etc..

Supervised Learning



Types of Supervised Learning



Regression

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

Classification

- Logistic Regression
- Spam Filtering
- Random Forest
- Decision Trees
- Support vector Machines

Regression:

A regression model predicts a numeric value. For example, a weather model that predicts the amount of rain, in inches or millimeters, is a regression model.

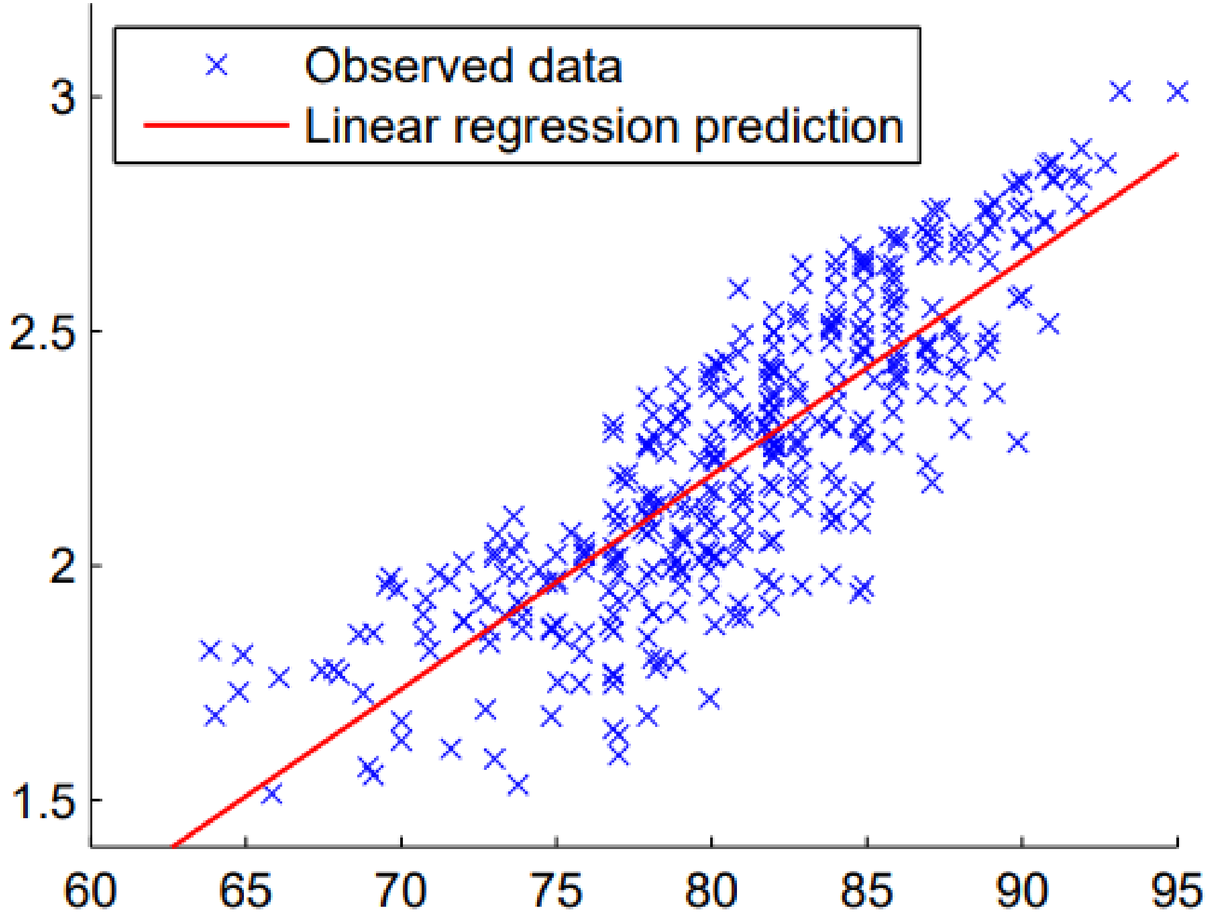
Classification:

Classification models predict the likelihood that something belongs to a category. Unlike regression models, whose output is a number, classification models output a value that states whether or not something belongs to a particular category. For example, classification models are used to predict if an email is spam or if a photo contains a cat.

Linear Regression

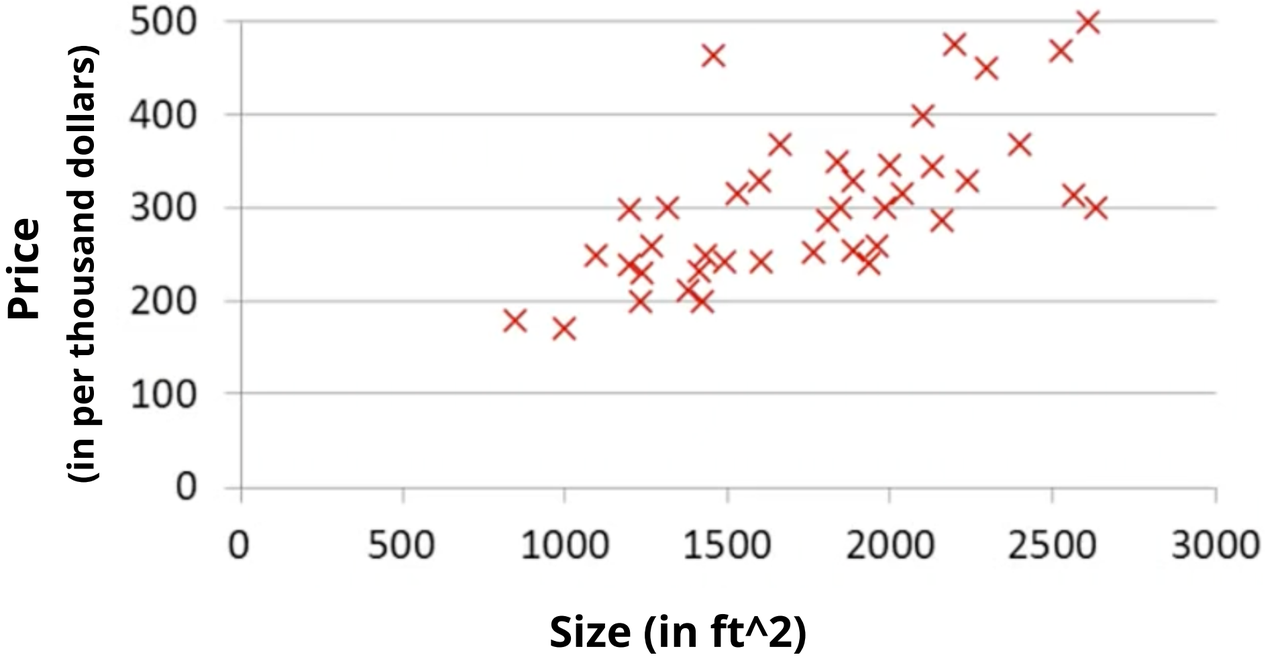
Linear Regression

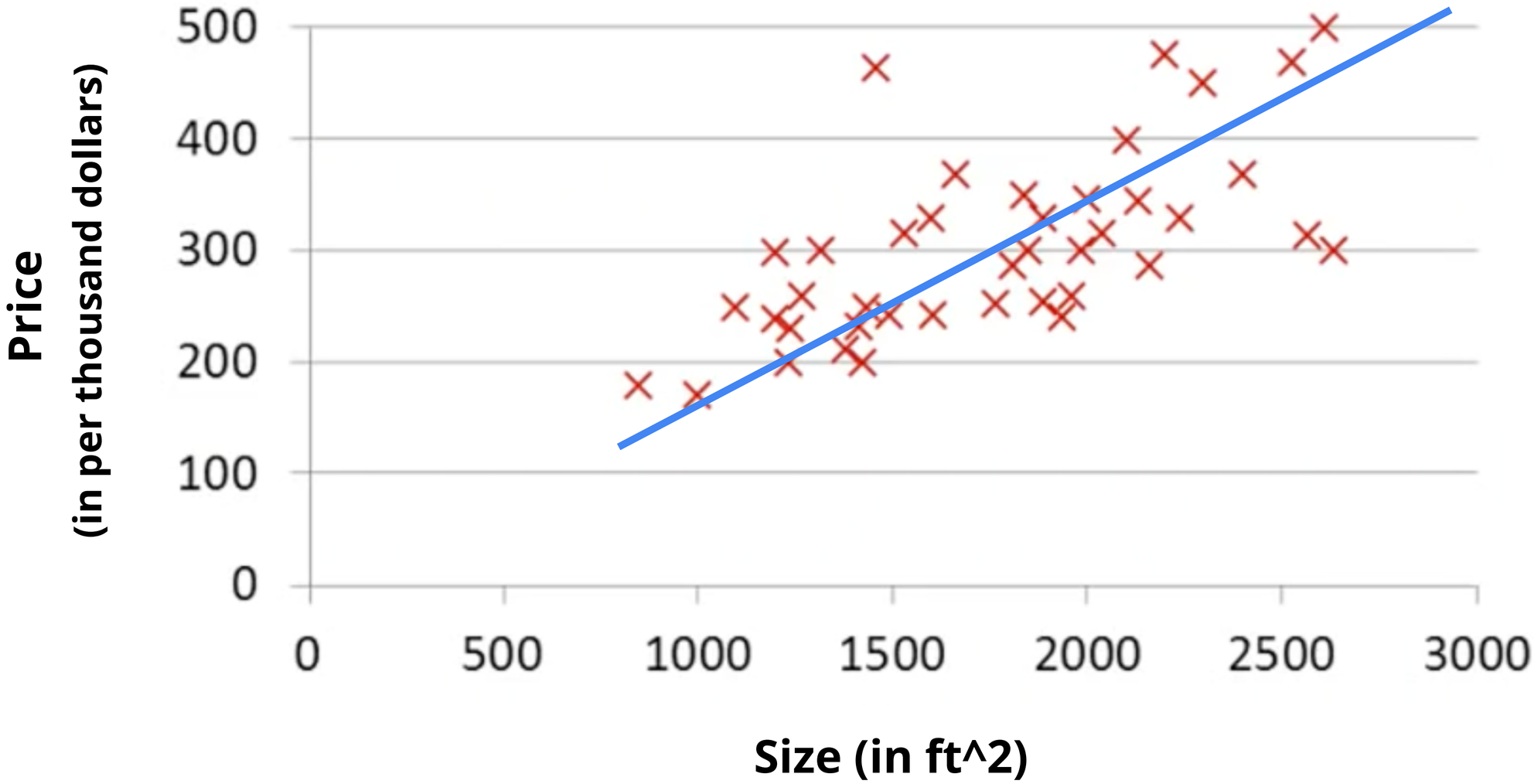
Linear Regression is the supervised Machine Learning model in which the model finds the best fit linear line between the independent and dependent variable i.e it finds the linear relationship between the dependent and independent variable.



Let's take an example..

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

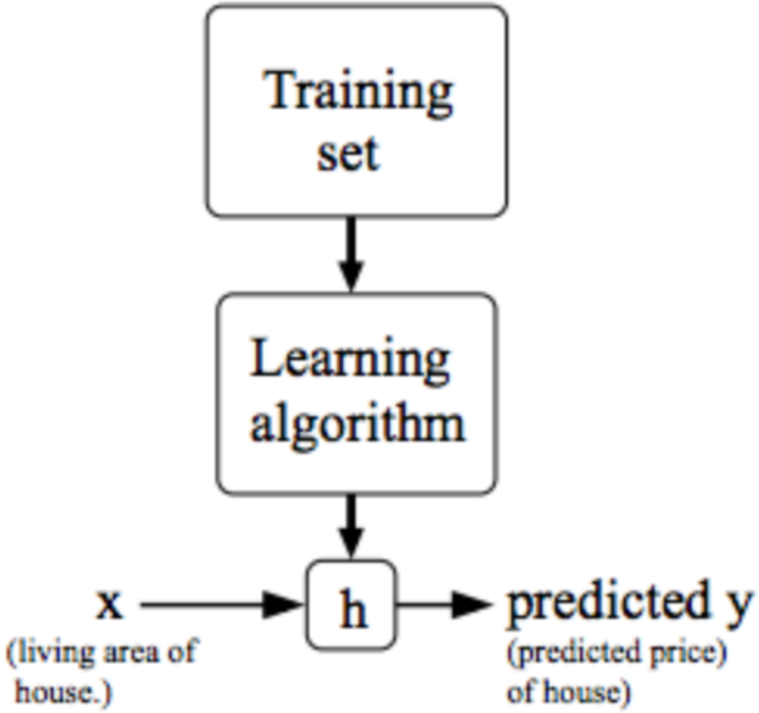



Price
(in per thousand dollars)
500
400
300
200
100
0
0
500
1000
1500
2000
2500
3000
Size (in ft^2)

Let us denote the training set, that we have, with the following notation:

$$(x^{(i)}, y^{(i)}); i = 1, ..., m$$

where, **m** = number of training examples we have in our dataset



Hypothesis function: To describe the supervised learning problem slightly more formally, our goal is, given a training set, to learn a function h : X → Y. And this function h is known as the hypothesis function.

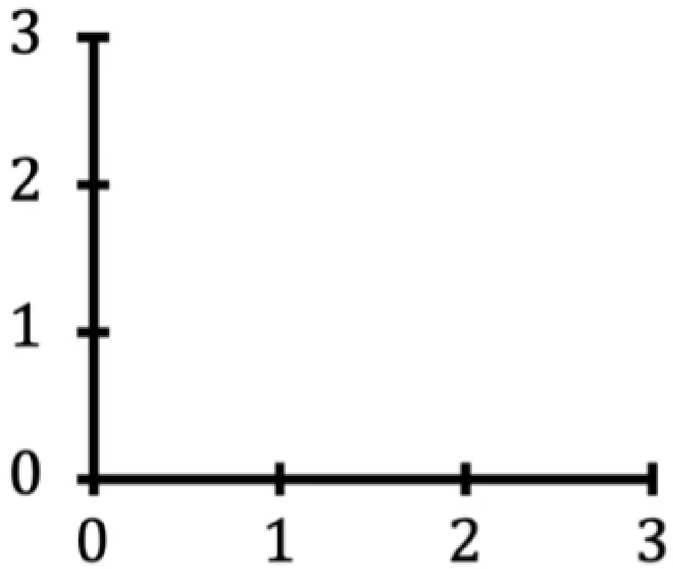
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

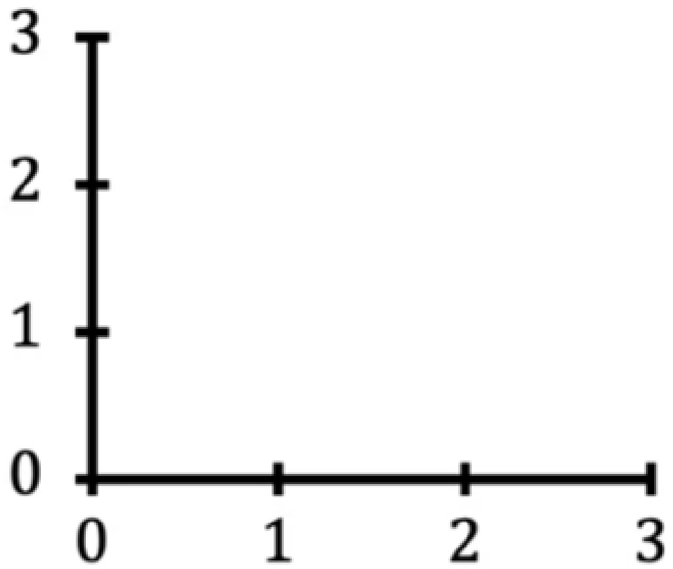
Parameters: θ_0, θ_1

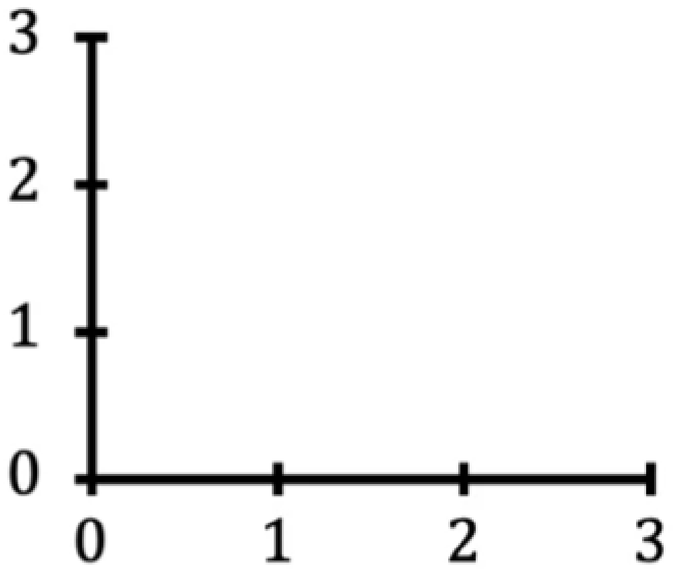
Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

$$h_\theta(x) = \theta_0 + \theta_1 x$$



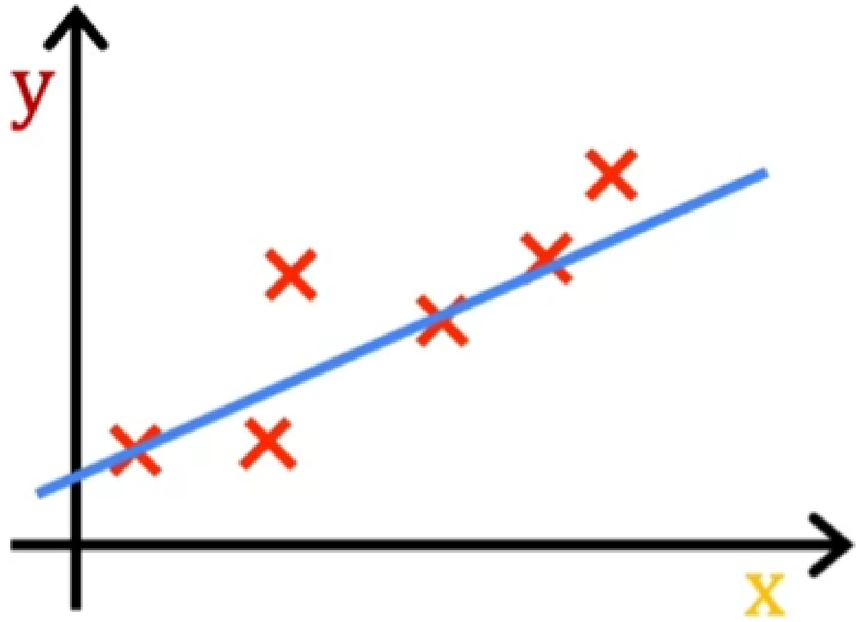




Error = $\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Total Error = $\sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

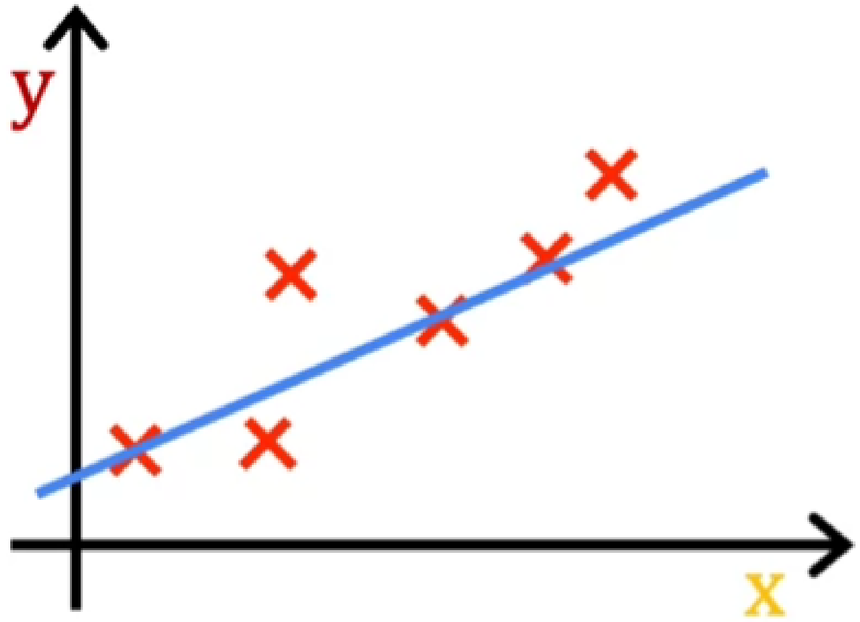
Mean Error = $\frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$



Error = $\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Total Error = $\sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Mean Error = $\frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$



Cost Function = $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

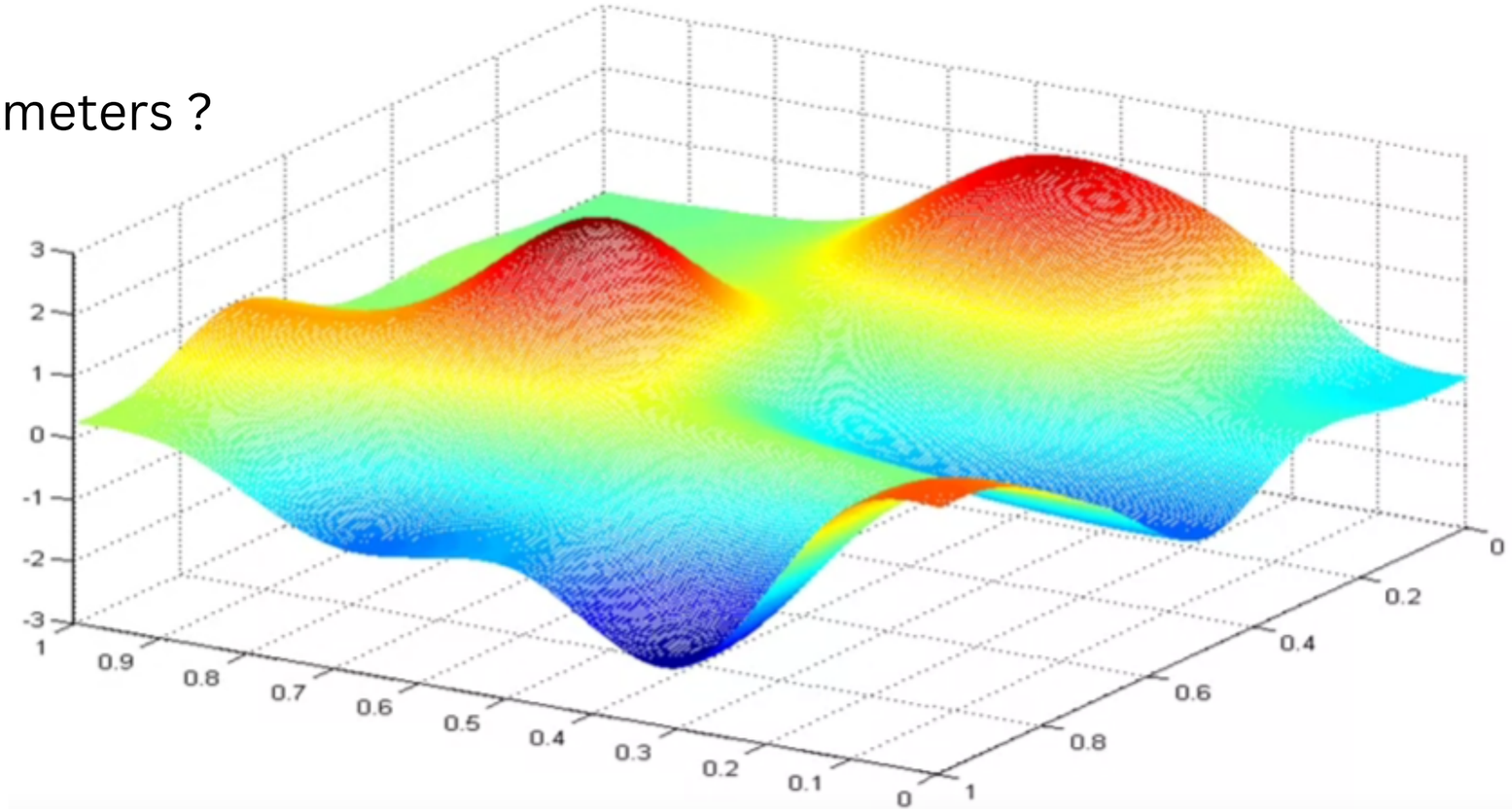
- Our Goal now is to minimise the error. i.e. to minimise the Cost function.
- We need to find the perfect parameters(θ_i) such that the mean error is minimum.

- Our Goal now is to minimise the error. i.e. to minimise the Cost function.
- We need to find the perfect parameters(θ_i) such that the mean error is minimum.

- How to find the perfect parameters ?

- Our Goal now is to minimise the error. i.e. to minimise the Cost function.
- We need to find the perfect parameters(θ_i) such that the mean error is minimum.

- How to find the perfect parameters ?



Have some function $J(\theta_0, \theta_1)$

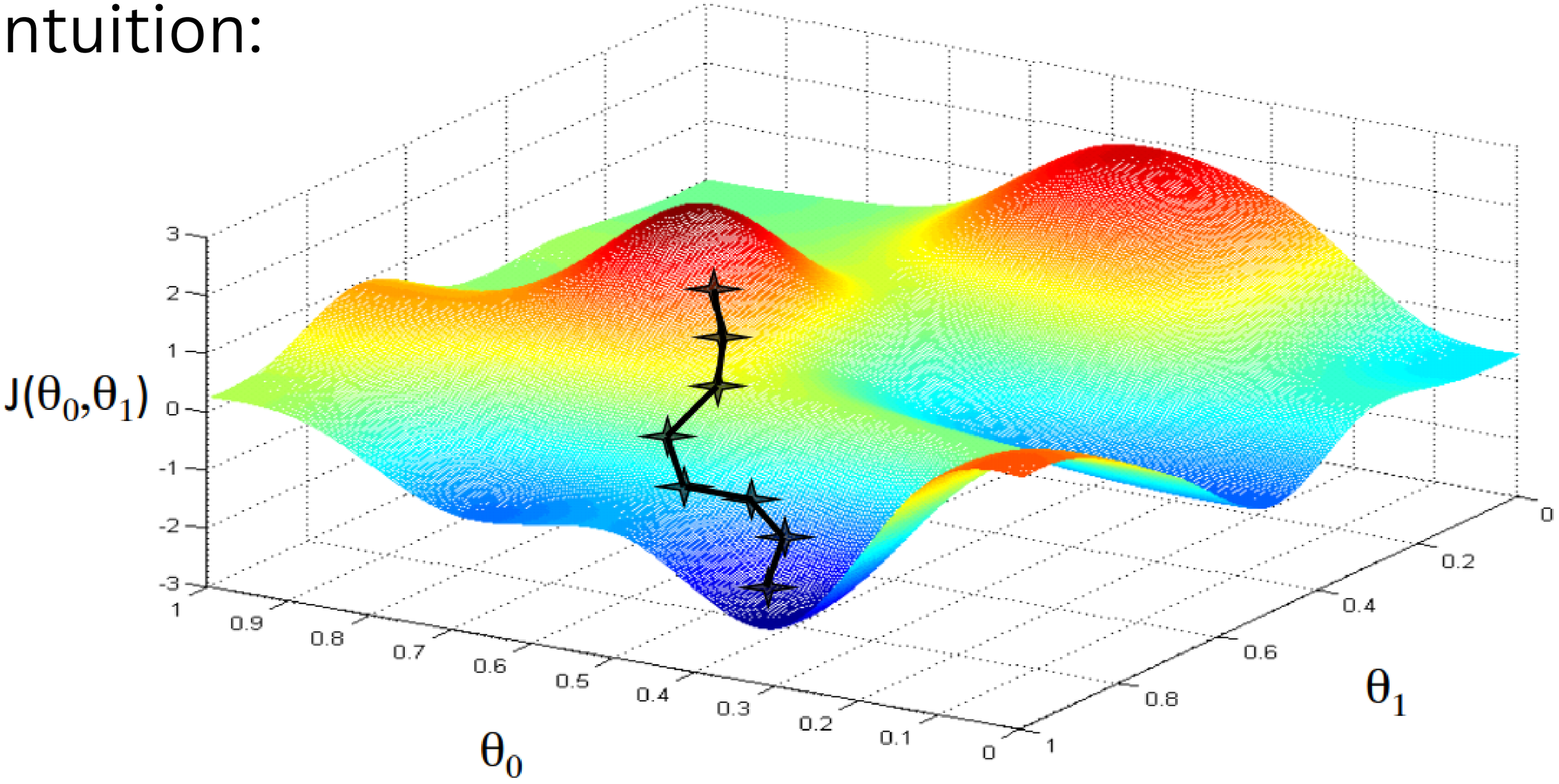
Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$

  until we hopefully end up at a minimum

Intuition:


$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

Gradient Descent Algorithm: $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(for $j = 1$ and $j = 0$)

}

Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \qquad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta(x) - y \right)^2 \\
&= 2 \cdot \frac{1}{2} \left(h_\theta(x) - y \right) \cdot \frac{\partial}{\partial \theta_j} \left(h_\theta(x) - y \right) \\
&= \left(h_\theta(x) - y \right) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right) \\
&= \left(h_\theta(x) - y \right) x_j
\end{aligned}$$

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(for $j = 1$ and $j = 0$)

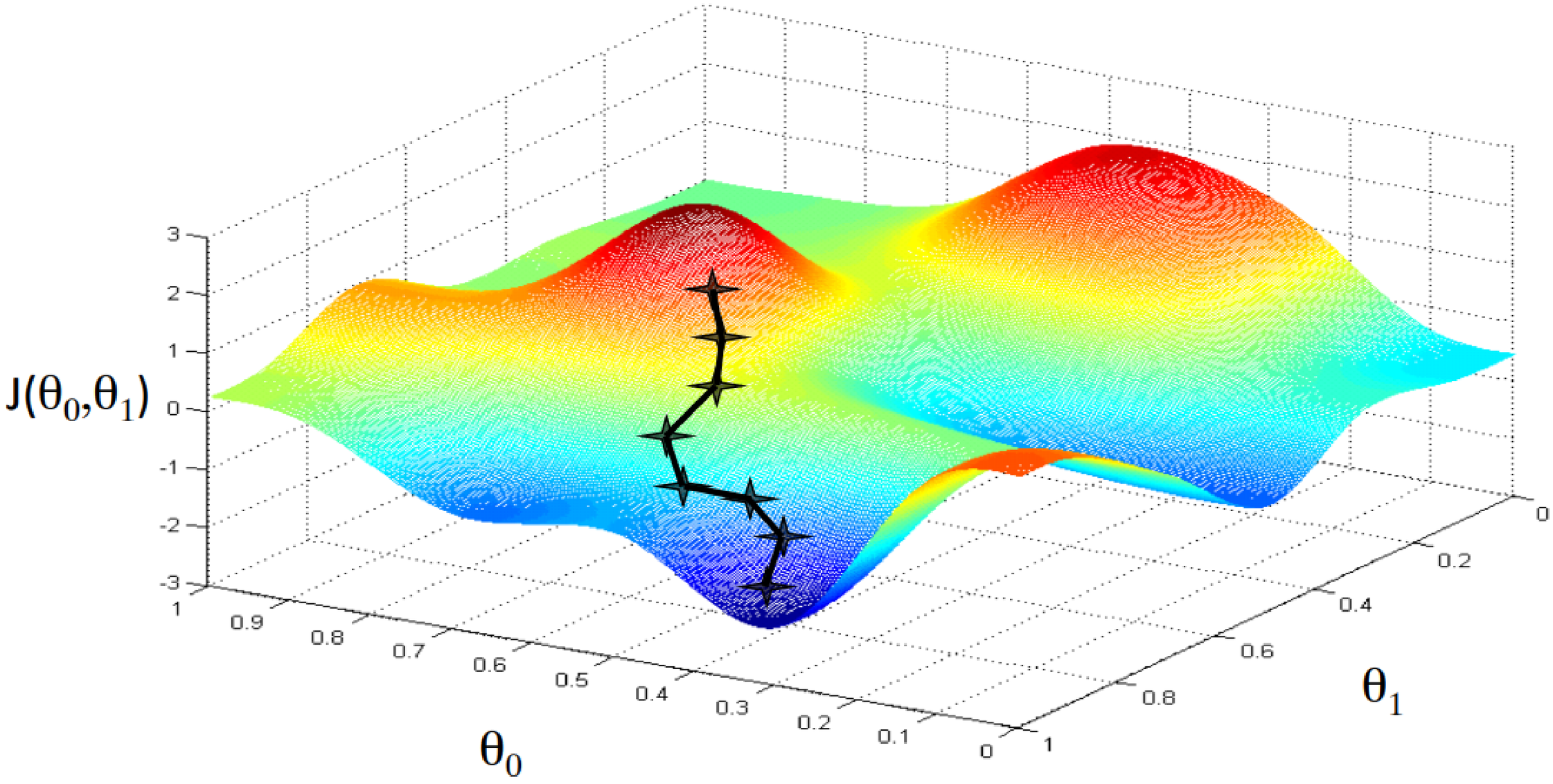
}

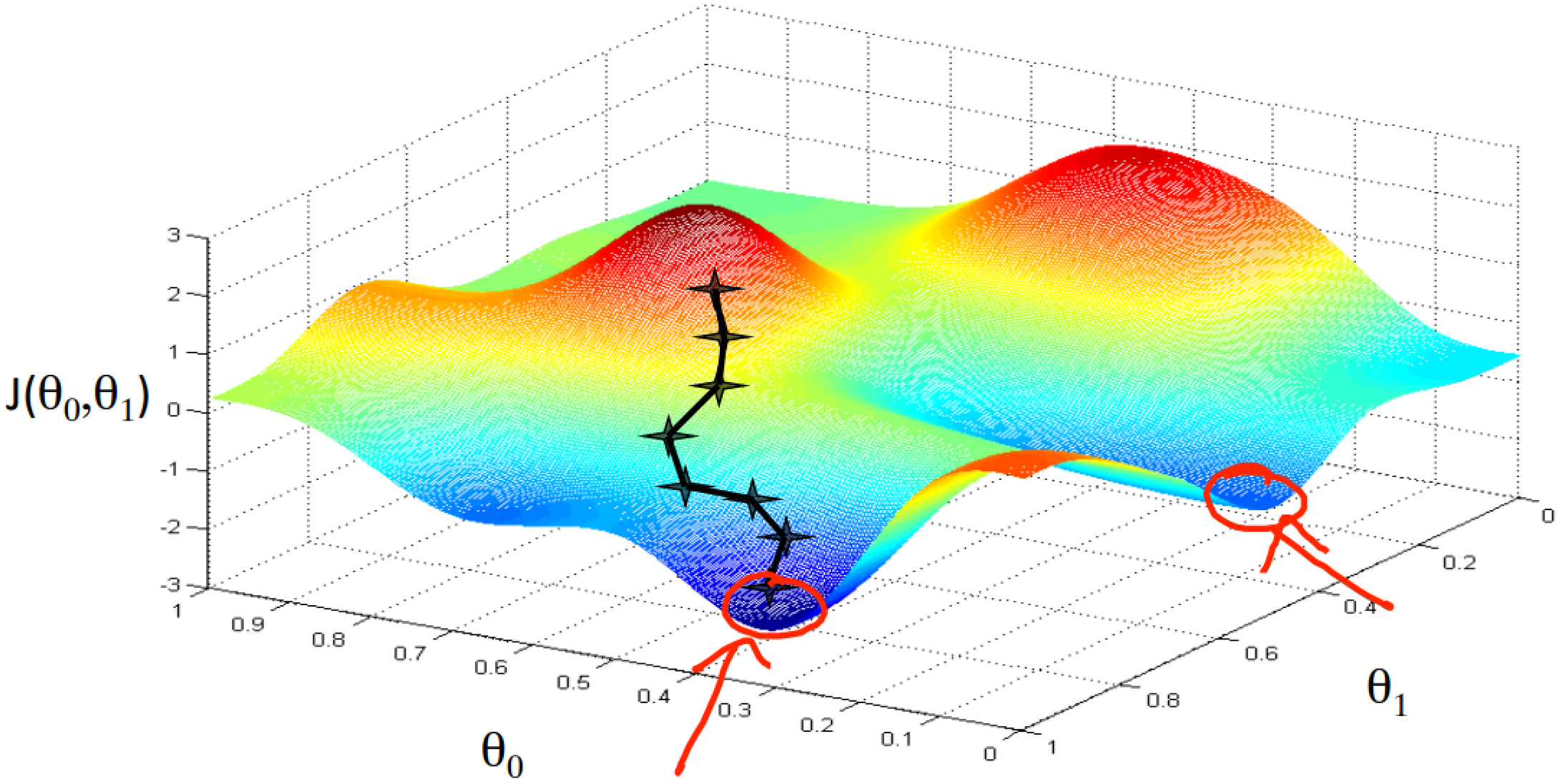
repeat until convergence: {

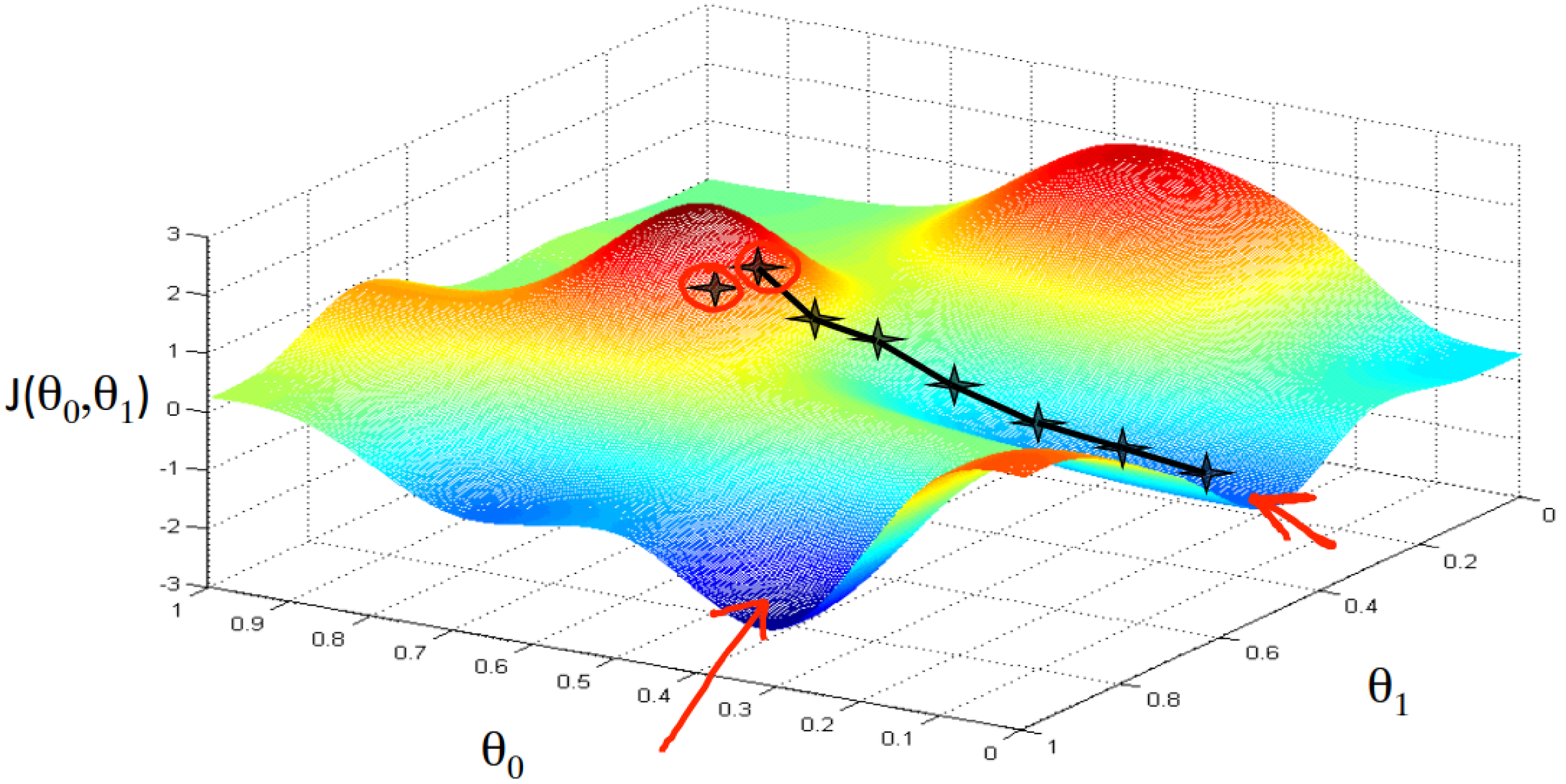
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

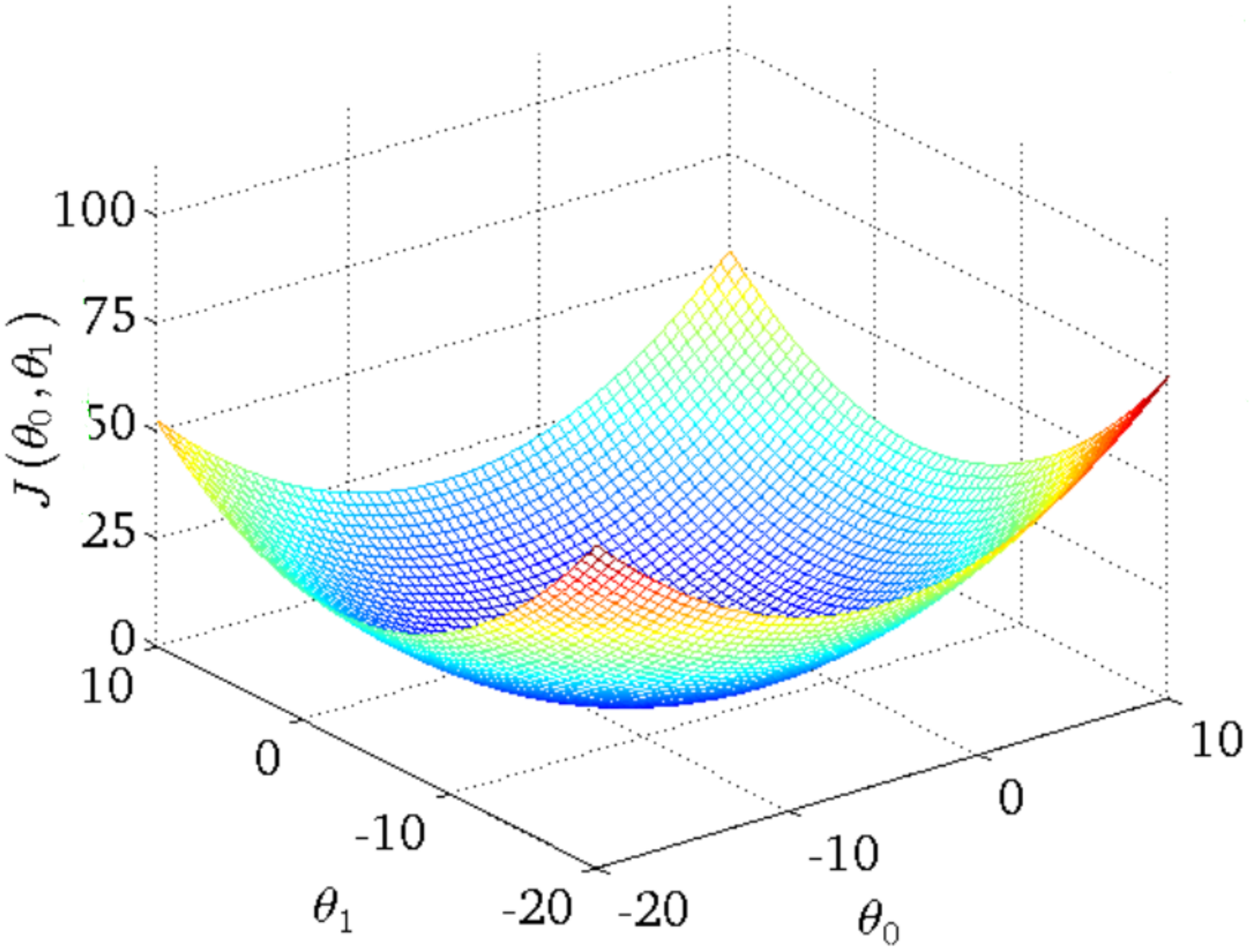
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left((h_\theta(x_i) - y_i) x_i \right)$$

}


$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1


J(θ0,θ1)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1


$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1


100
75
50
25
0
$J(\theta_0, \theta_1)$
10
0
-10
-20
θ_1
-20
-10
0
10
θ_0

https://lukaszkujawa.github.io/gradient-descent.html

Multiple Features

Linear regression with multiple variables is also known as "multivariate linear regression".

The multivariable form of the hypothesis function accommodating these multiple features is as follows:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

$$h_\theta(x) = \begin{bmatrix} \theta_0 & \theta_1 & \dots & \theta_n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \theta^T x$$

Multiple Features

Size in feet^2	Number of bedrooms	Number of floors	Age of home in years	Price ($) in $1000's
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

Gradient Descent for Multiple Variables

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

- The gradient descent equation itself is generally the same form; we just have to repeat it for our 'n' features:

$$\begin{aligned}
&\text{repeat until convergence: } \{ \\
&\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)} \\
&\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)} \\
&\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)} \\
&\cdots \\
&\}
\end{aligned}$$

In other words:

repeat until convergence: {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \qquad \text{for } j := 0...n$$

}

$x_j^{(i)}$ = value of feature j in the i^{th} training example

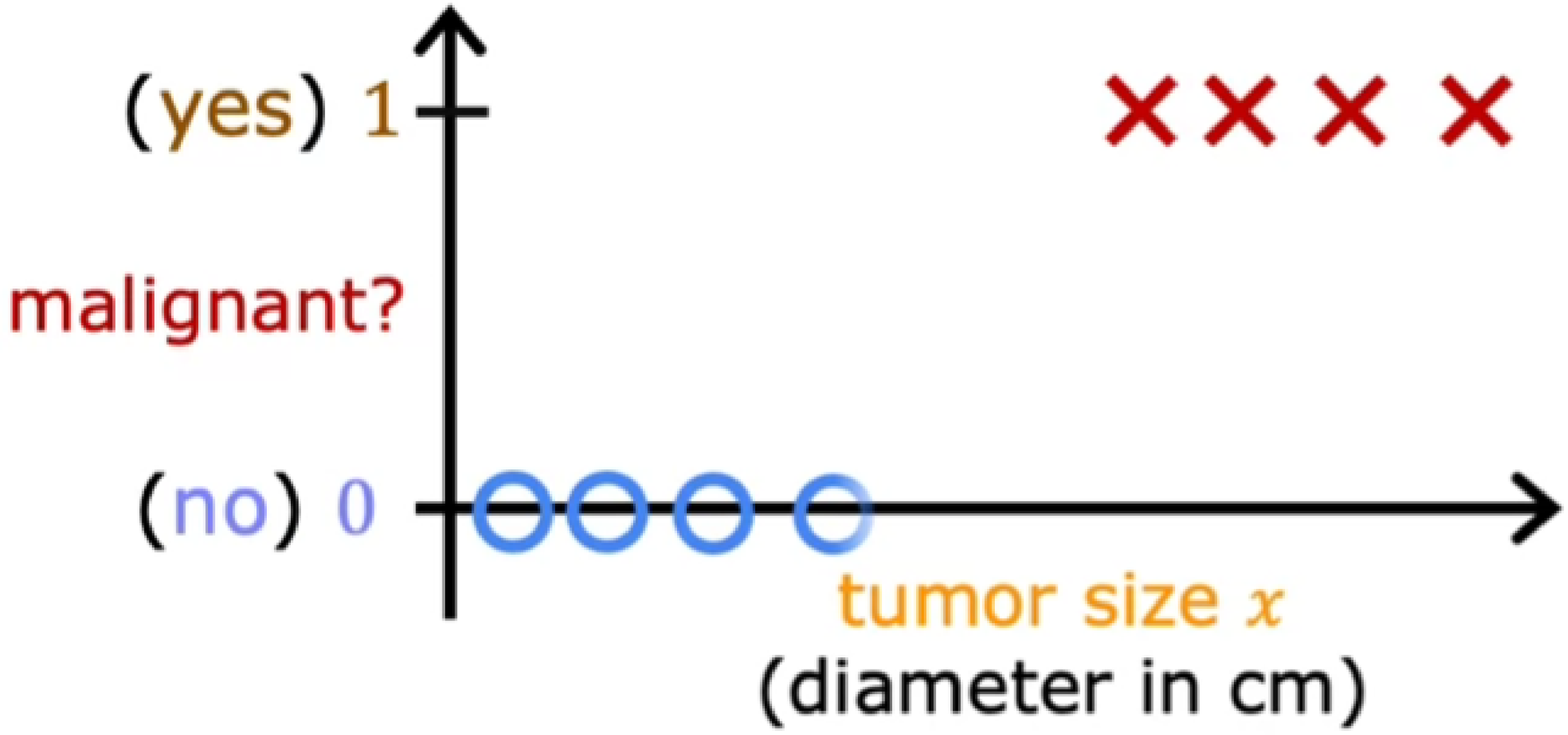
$x^{(i)}$ = the input (features) of the i^{th} training example

m = the number of training examples

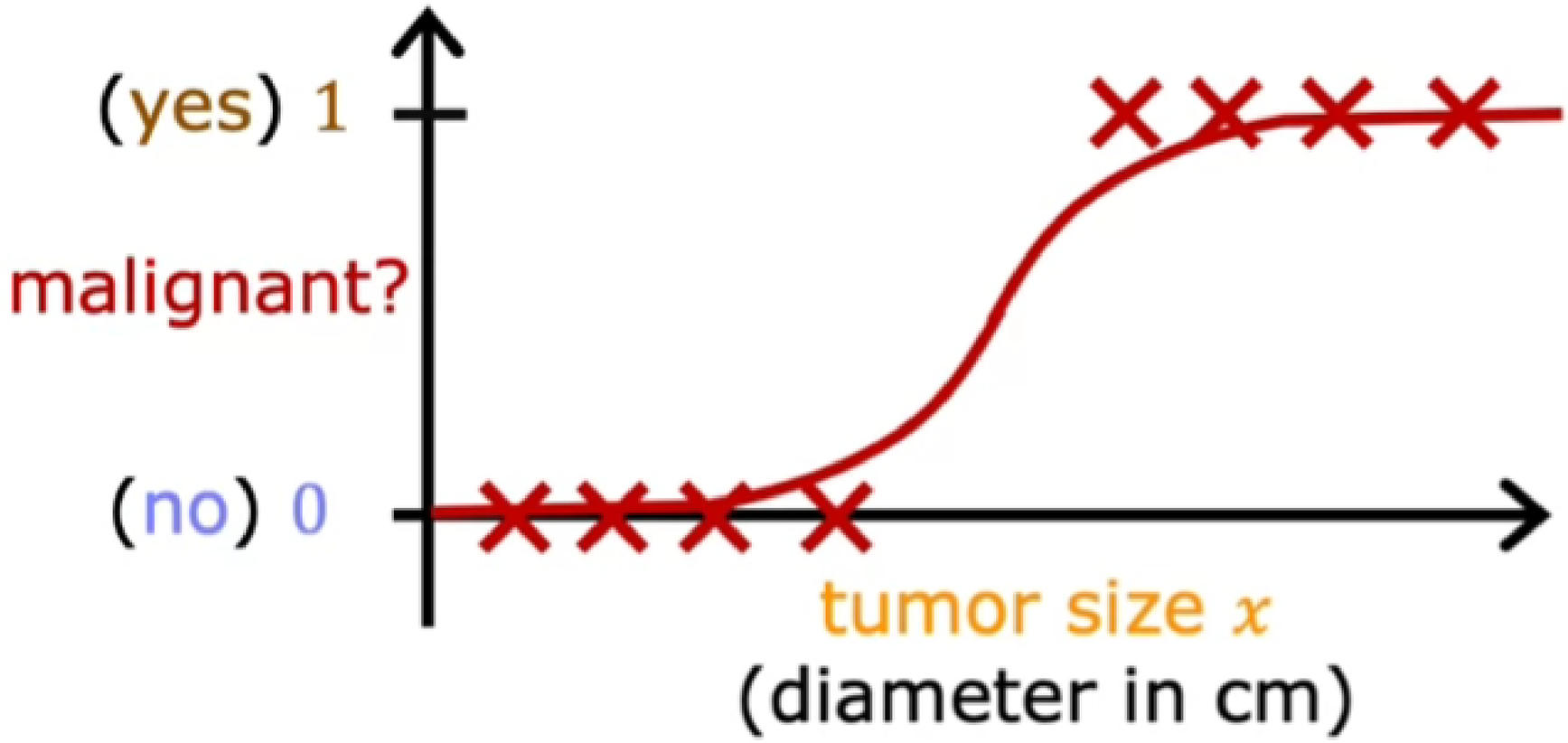
n = the number of features

Logistic Regression

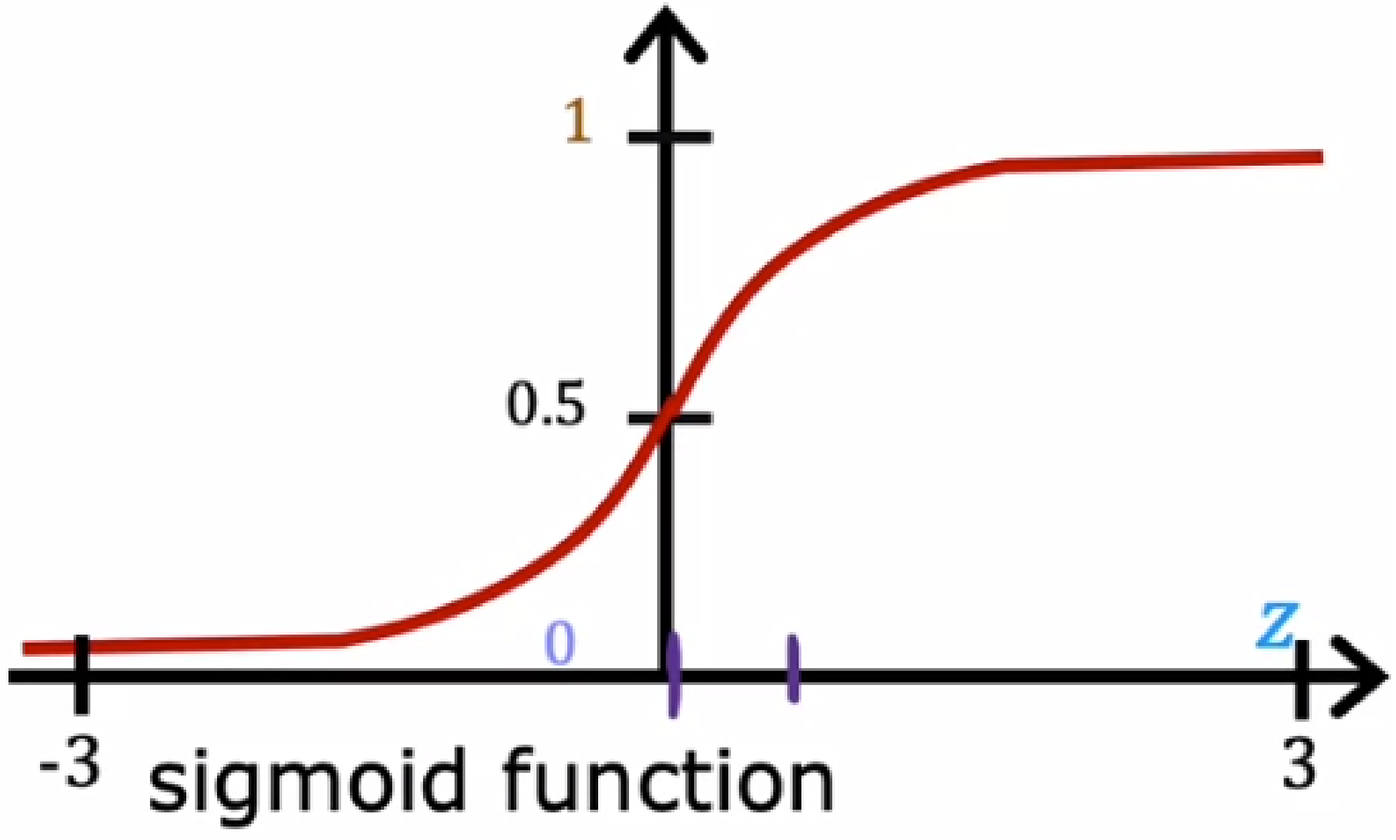
Motivation:



Motivation:

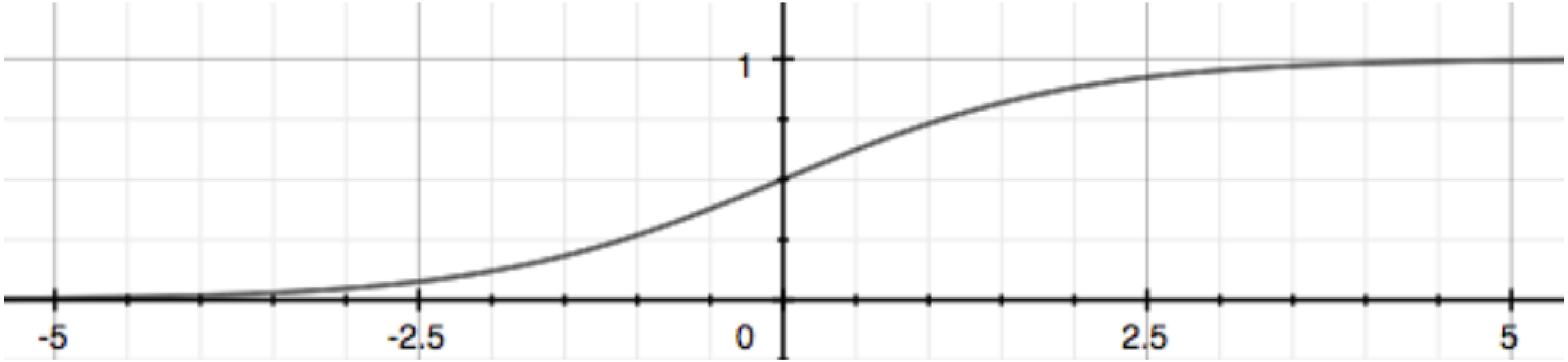


Want outputs between 0 and 1



sigmoid function

logistic function



$$g(z) = \frac{1}{1 + e^{-z}}$$

We use the "Sigmoid Function," also called the "Logistic Function":

Want $0 \leq h_\theta(x) \leq 1$

we have, $g(z) = \frac{1}{1 + e^{-z}}$ ⟶ g(z) outputs a value between 0 and 1

$$h_\theta(x) = g(\theta^T x)$$

$h_\theta(x)$ = estimated probability that y = 1 on input x

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$$h_\theta(x) = 0.7$$

Tell patient that 70% chance of tumor being malignant

Decision Boundary

$$h_\theta(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1+e^{-z}}$$

Suppose predict "$y = 1$" if $h_\theta(x) \geq 0.5$

predict "$y = 0$" if $h_\theta(x) < 0.5$

$g(z) \geq 0.5$
when $z \geq 0$

The way our logistic function g behaves is that when its input is greater than or equal to zero, its output is greater than or equal to 0.5:

Remember:

$$z = 0, e^0 = 1 \Rightarrow g(z) = 1/2$$
$$z \to \infty, e^{-\infty} \to 0 \Rightarrow g(z) = 1$$
$$z \to -\infty, e^{\infty} \to \infty \Rightarrow g(z) = 0$$

So if our input to g is $\theta^T X$, then that means:

$$h_\theta(x) = g(\theta^T x) \geq 0.5$$
$$\text{when } \theta^T x \geq 0$$

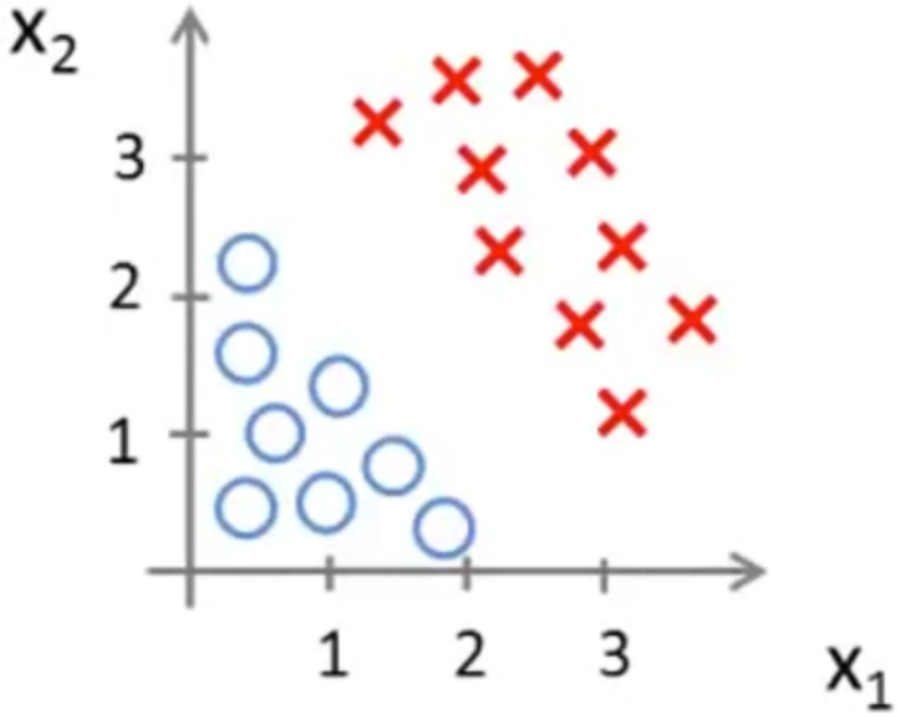
From these statements we can now say:

$$\theta^T x \geq 0 \Rightarrow y = 1$$
$$\theta^T x < 0 \Rightarrow y = 0$$

The decision boundary is the line that separates the area where y = 0 and where y = 1. It is created by our hypothesis function.

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

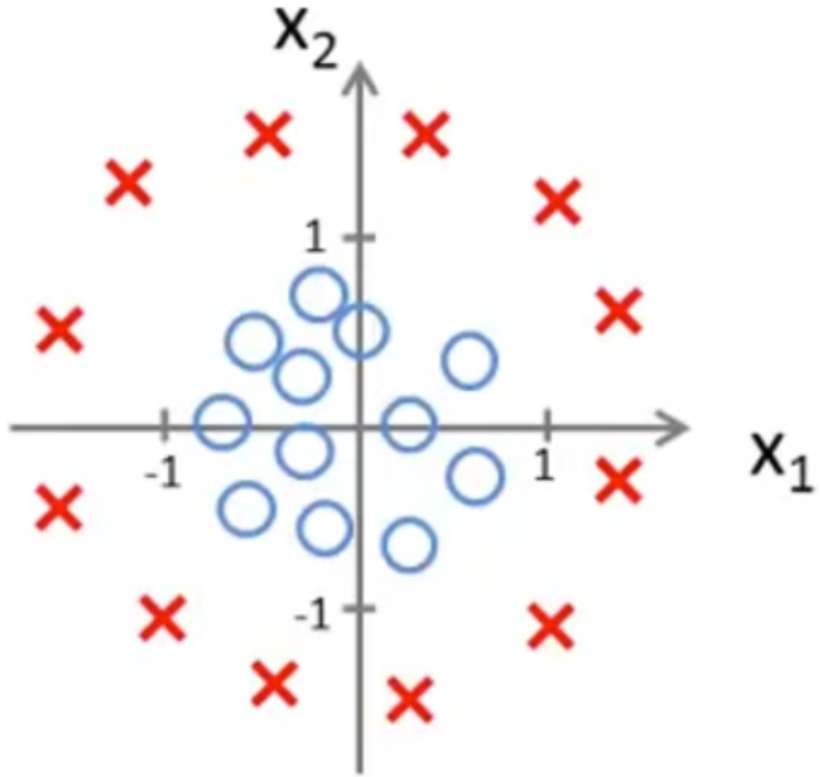
Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$



The decision boundary can be non-linear also:

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$



Cost Function:

- We cannot use the same cost function that we use for linear regression because the Logistic Function will cause the output to be wavy, causing many local optima. In other words, it will not be a convex function.

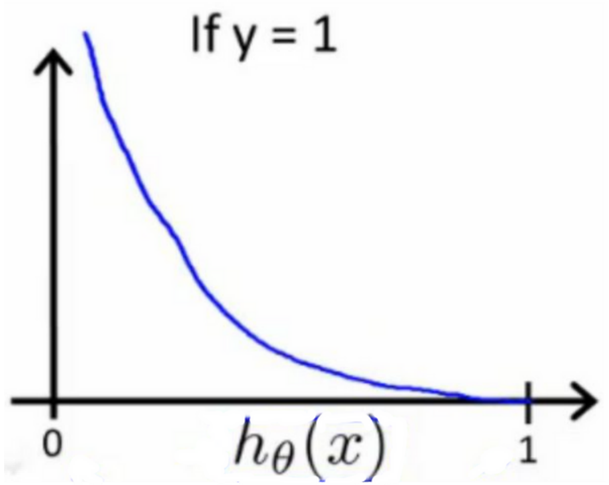
- Instead, our cost function for logistic regression looks like:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

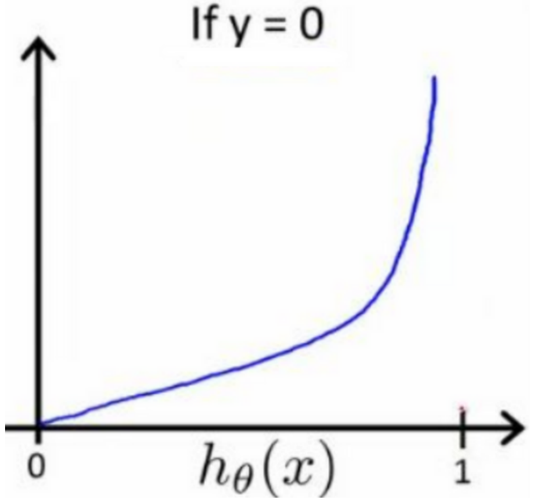
$$\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \qquad \text{if } y = 1$$

$$\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \qquad \text{if } y = 0$$

When y = 1, we get the following plot for $J(\theta)$ vs $h_\theta(x)$:



Similarly, when y = 0, we get the following plot for $J(\theta)$ vs $h_\theta(x)$:



$$\text{Cost}(h_\theta(x), y) = 0 \text{ if } h_\theta(x) = y$$
$$\text{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 0 \text{ and } h_\theta(x) \to 1$$
$$\text{Cost}(h_\theta(x), y) \to \infty \text{ if } y = 1 \text{ and } h_\theta(x) \to 0$$

- If our correct answer 'y' is 0, then the cost function will be 0 if our hypothesis function also outputs 0. If our hypothesis approaches 1, then the cost function will approach infinity.

- If our correct answer 'y' is 1, then the cost function will be 0 if our hypothesis function outputs 1. If our hypothesis approaches 0, then the cost function will approach infinity.

Simplified Cost Function and Gradient Descent:

Our previous cost function was:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \qquad \text{if } y = 1$$

$$\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \qquad \text{if } y = 0$$

We can compress our cost function's two conditional cases into one case:

$$\text{Cost}(h_\theta(x), y) = -y\ \log(h_\theta(x)) - (1 - y)\log(1 - h_\theta(x))$$

Gradient Descent:

- We can fully write out our entire cost function as follows:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

- Remember that the general form of gradient descent is:

$$\text{Repeat } \{$$
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$
$$\}$$

We can work out the derivative part using calculus to get:

$$\text{Repeat } \{$$

$$\theta_j := \theta_j - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$\}$$

this algorithm is identical to the one we used in linear regression. We still have to simultaneously update all values in theta.

Unsupervised Learning

What is Unsupervised learning?

- In unsupervised learning, the model is trained on unlabeled data and tasked to discover patterns, structures, and relationships within the data.
- The most commonly used unsupervised learning algorithms include clustering, dimensionality reduction, and anomaly detection.
- Unsupervised learning is widely used in various applications, such as market segmentation, customer profiling, image compression, and genetic analysis.

1. Clustering

- Clustering algorithms divide the data into groups(clusters) based on similarity.
- Different types of clustering such as- Centroid based clustering, Density-based clustering, Distribution-based and Hierarchical Clustering.

2. Dimensionality Reduction

- Dimensionality reduction algorithms, such as PCA (Principal Component Analysis), transform high-dimensional data into a lower-dimensional space while preserving important relationships among the features.

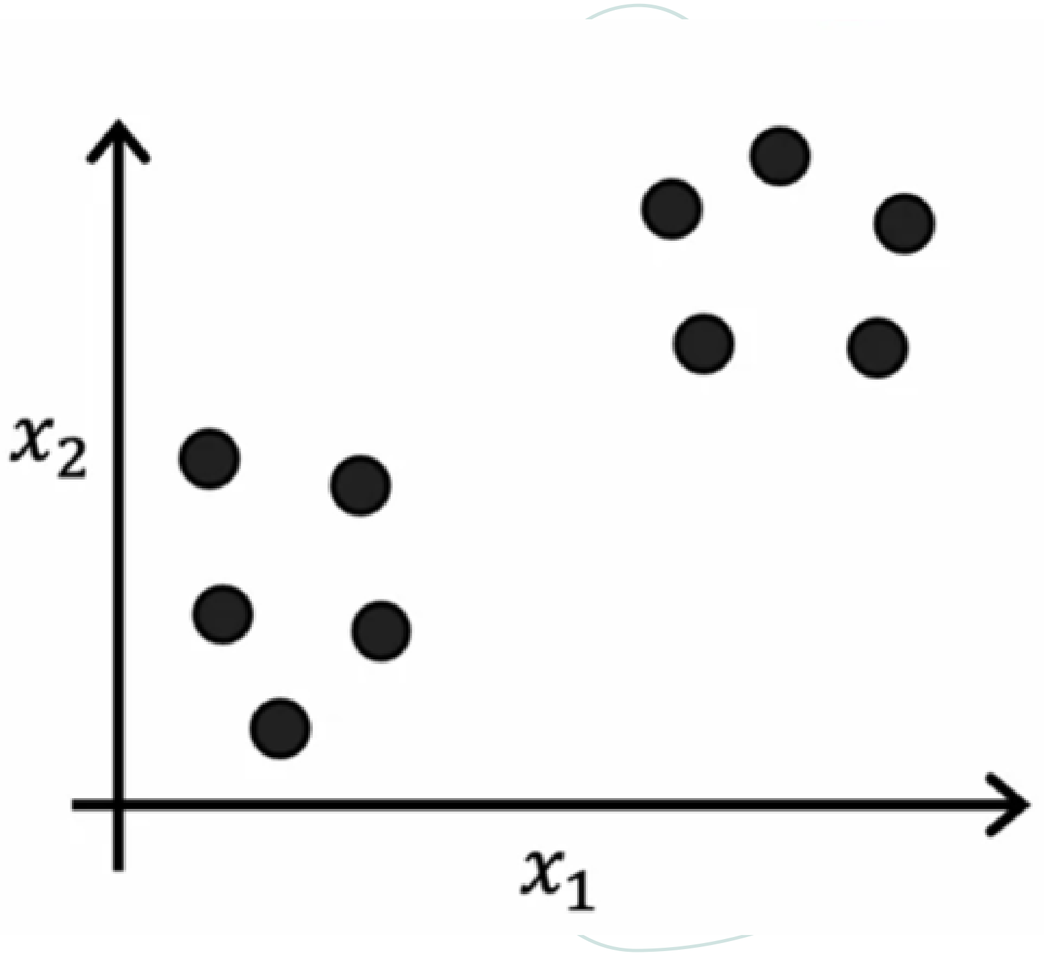
3. Anomaly Detection

- Anomaly detection is used to identify unusual or unexpected observations in a dataset.
- Flag data points that fall outside of the "normal" range.

Clustering

Types of CLustering:

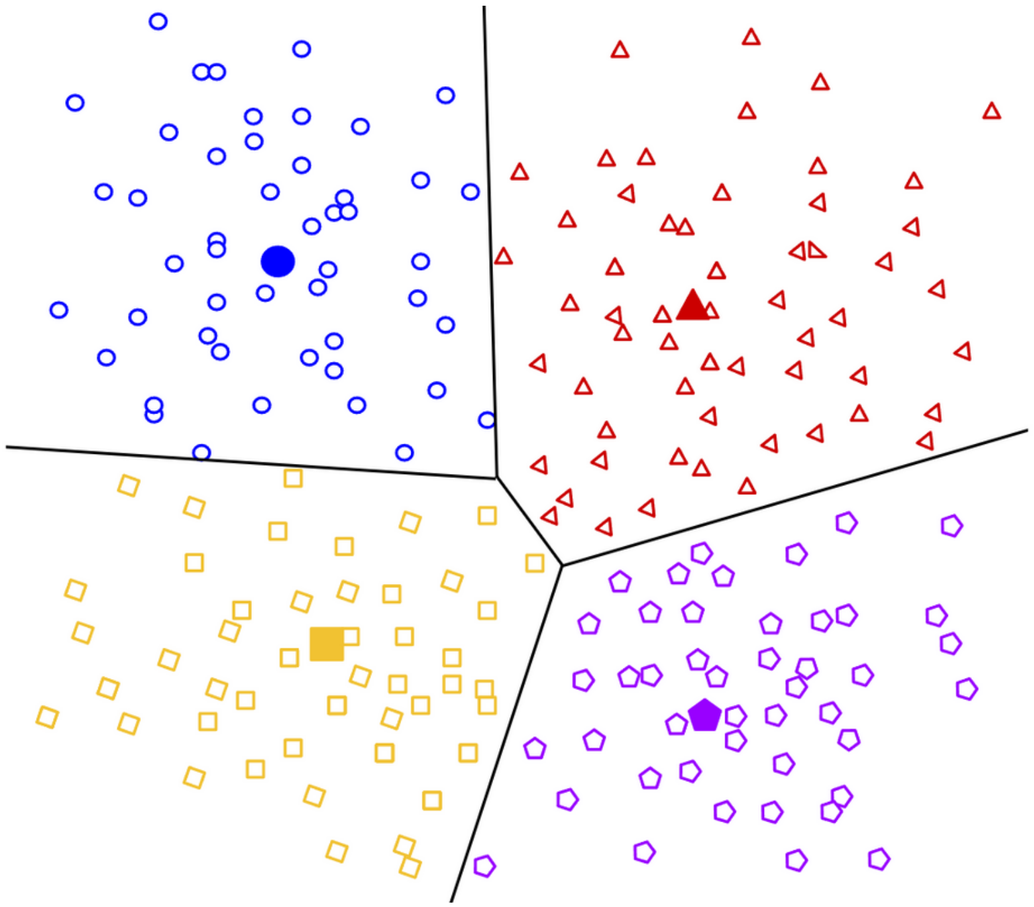
- Centroid-based Clustering
- Density-based Clustering
- Distribution-based Clustering
- Hierarchical Clustering



Centroid-based Clustering

Groups Data Points into Clusters with Centroids

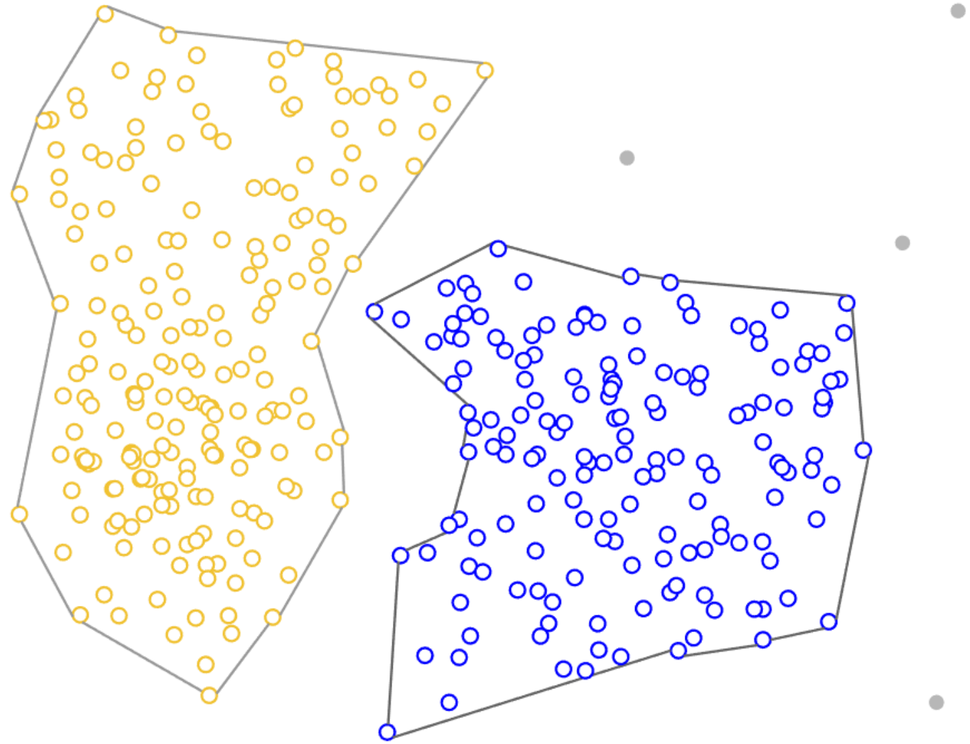
- Centroid-based clustering involves grouping data points into clusters based on the distances between each data point and the centroid of the cluster.
- Minimises the sum of squared distances between each data point and the centroid of its cluster.



Density-based Clustering

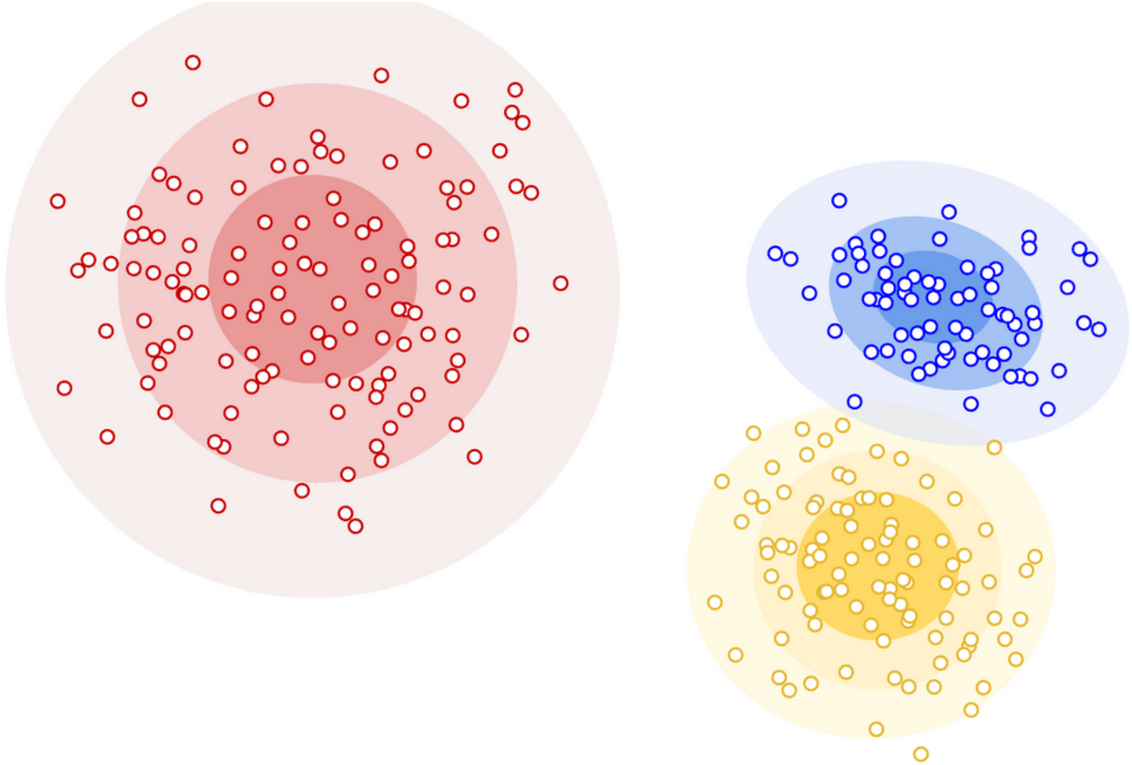
Connects areas of high density into clusters

- Does not require the user to specify the number of clusters a priori. Instead, it works by identifying dense areas in the dataset and grouping the data points within them into separate clusters.
- This is well-suited for datasets that have complex, non-linear shapes or contain noisy data points.
- Handles noise in data better than centroid-based clustering algorithms.



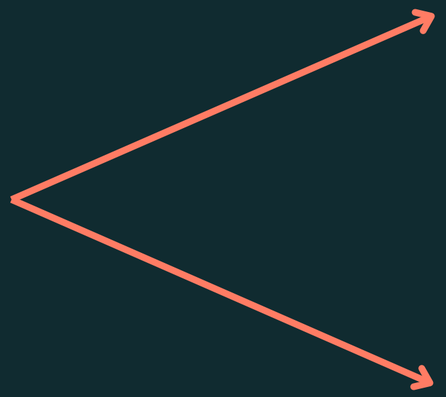
Distribution-based Clustering

Assumes data is composed of distributions, such as **Gaussian distributions**.

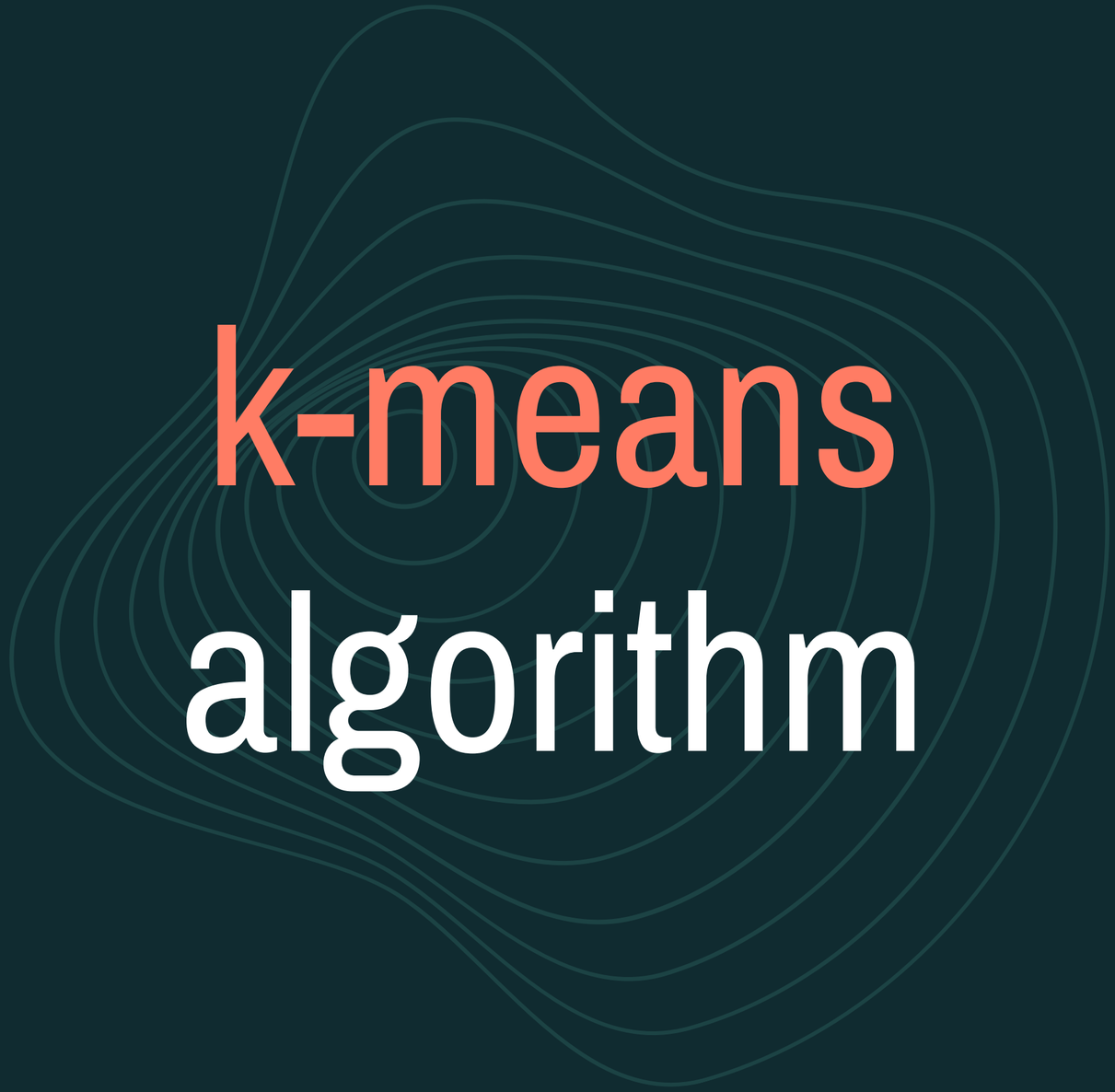


- In the figure, the data has been clustered into three gaussian distributions.
- As the distance from the distribution's centre increases, the probability that a point belongs to the distribution decreases.
- This is useful when your data has a particular distribution.

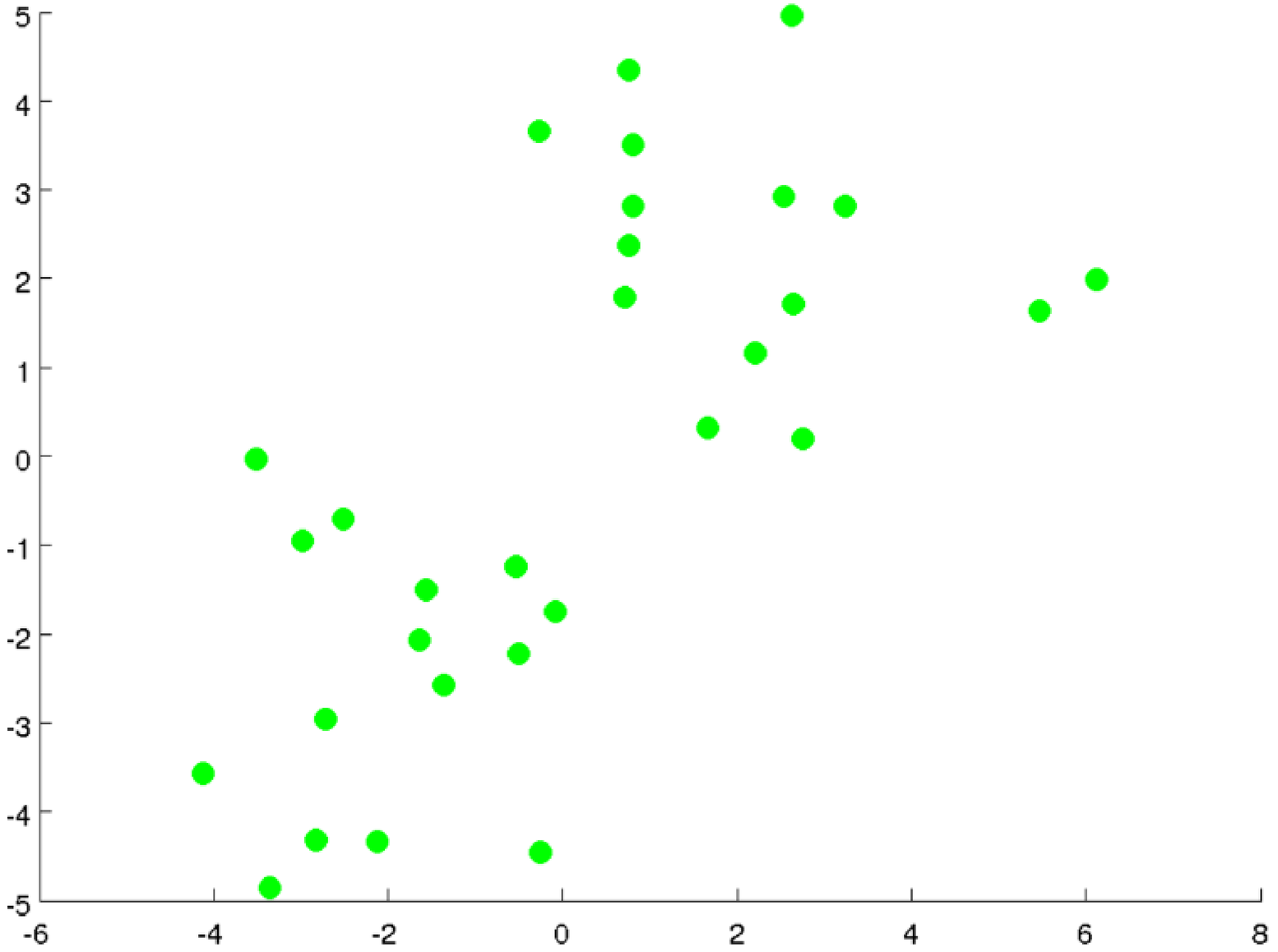
Centroid-based Clustering

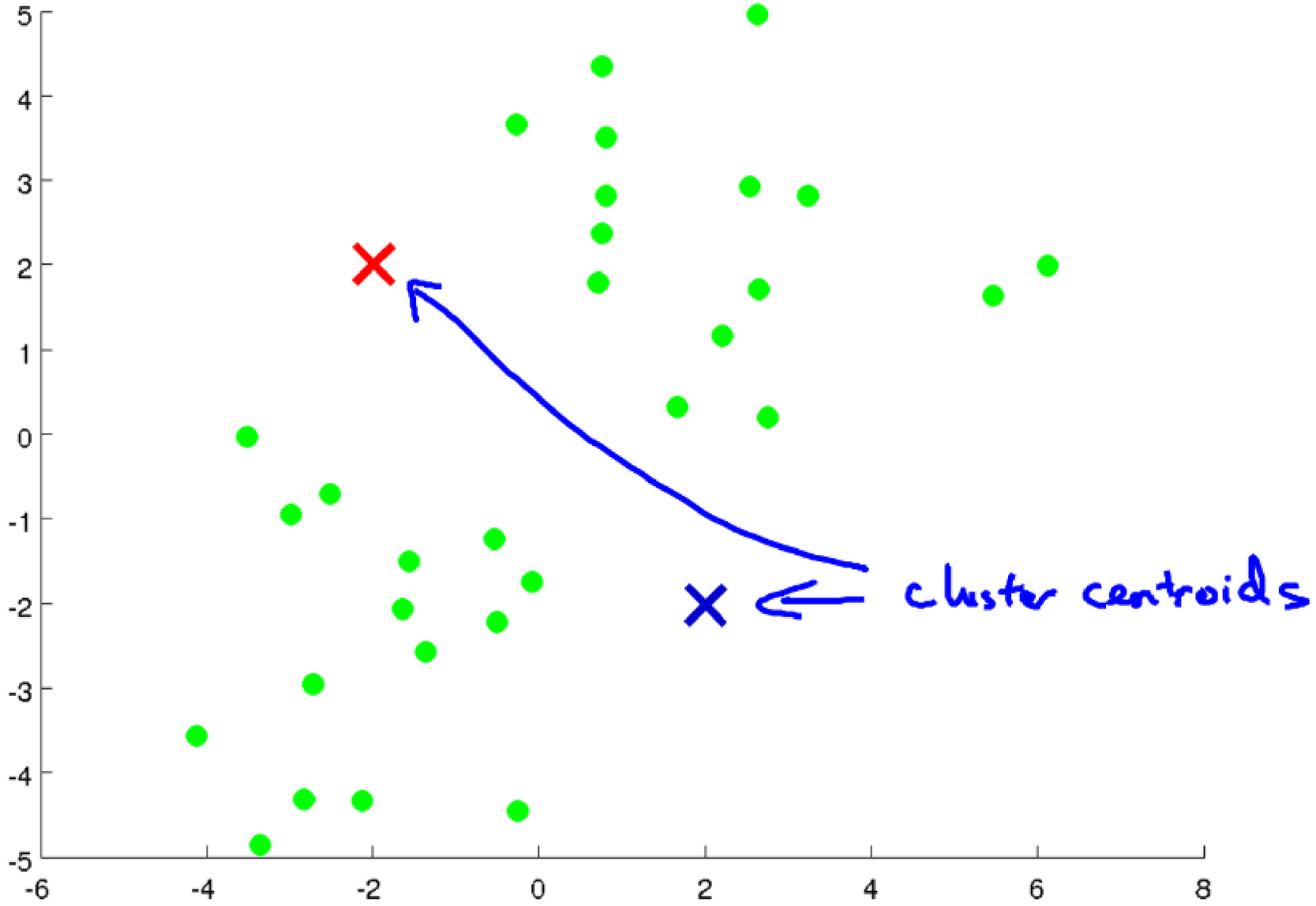


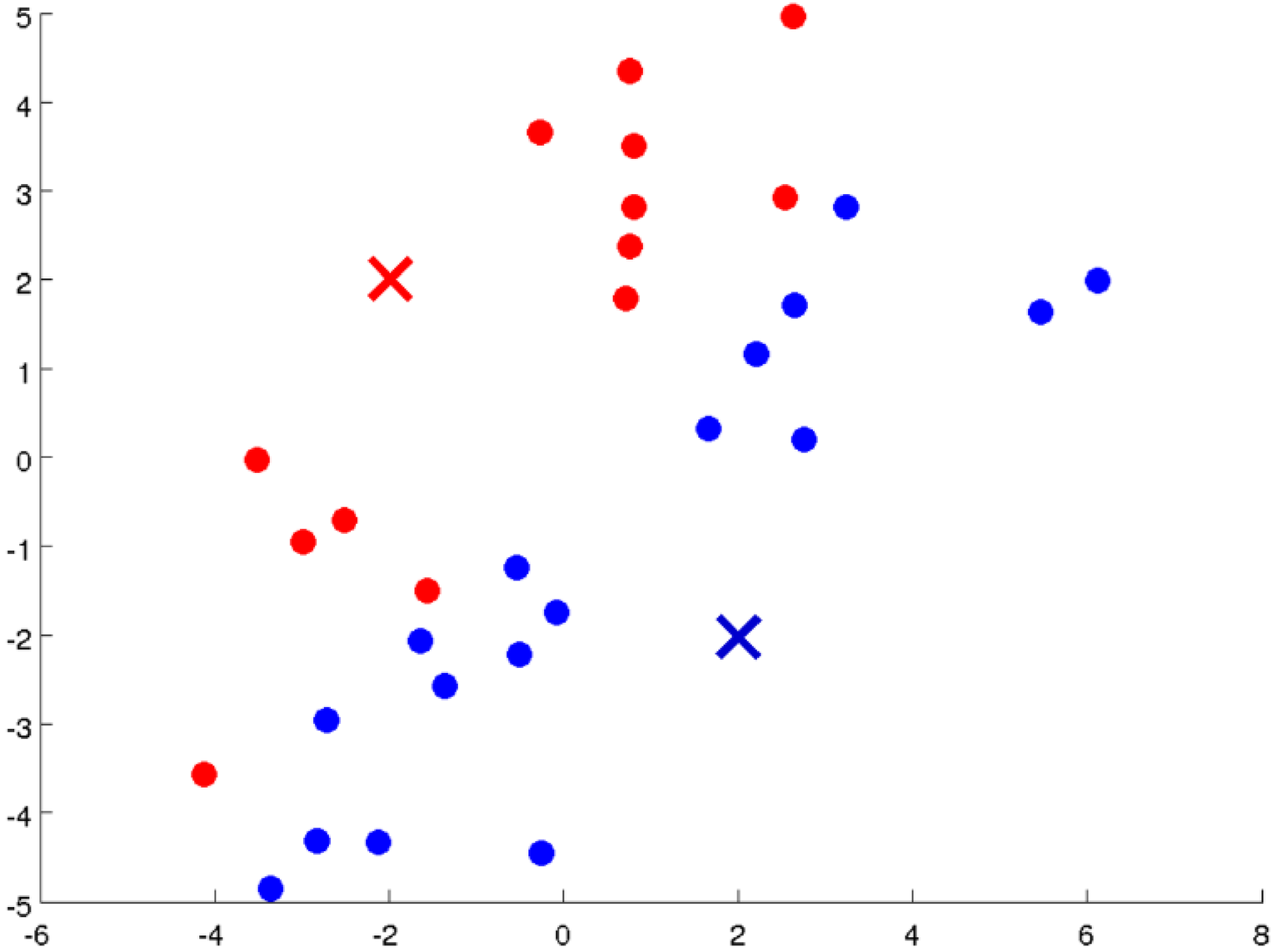
- k-means
- k-nearest neighbours

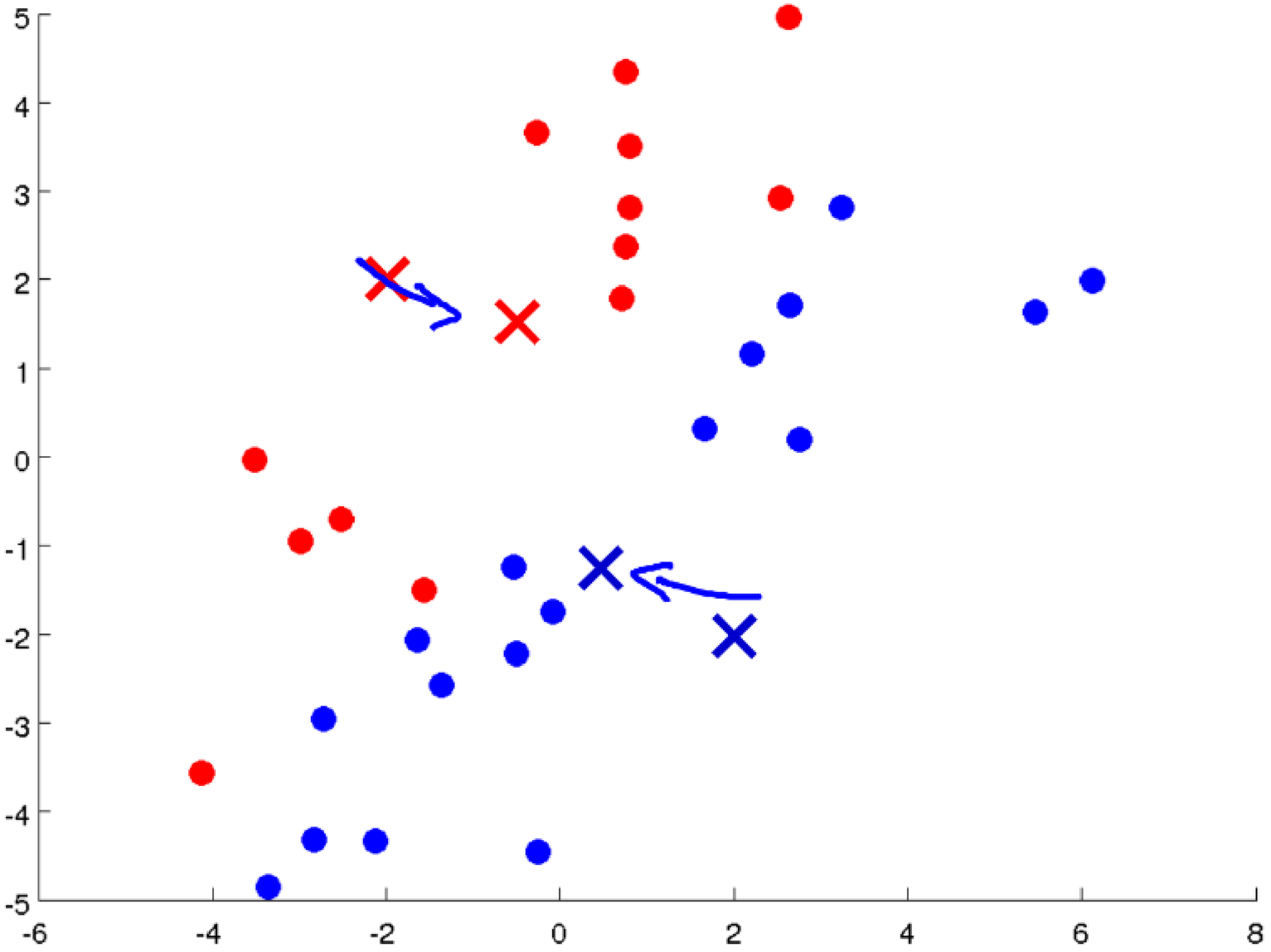


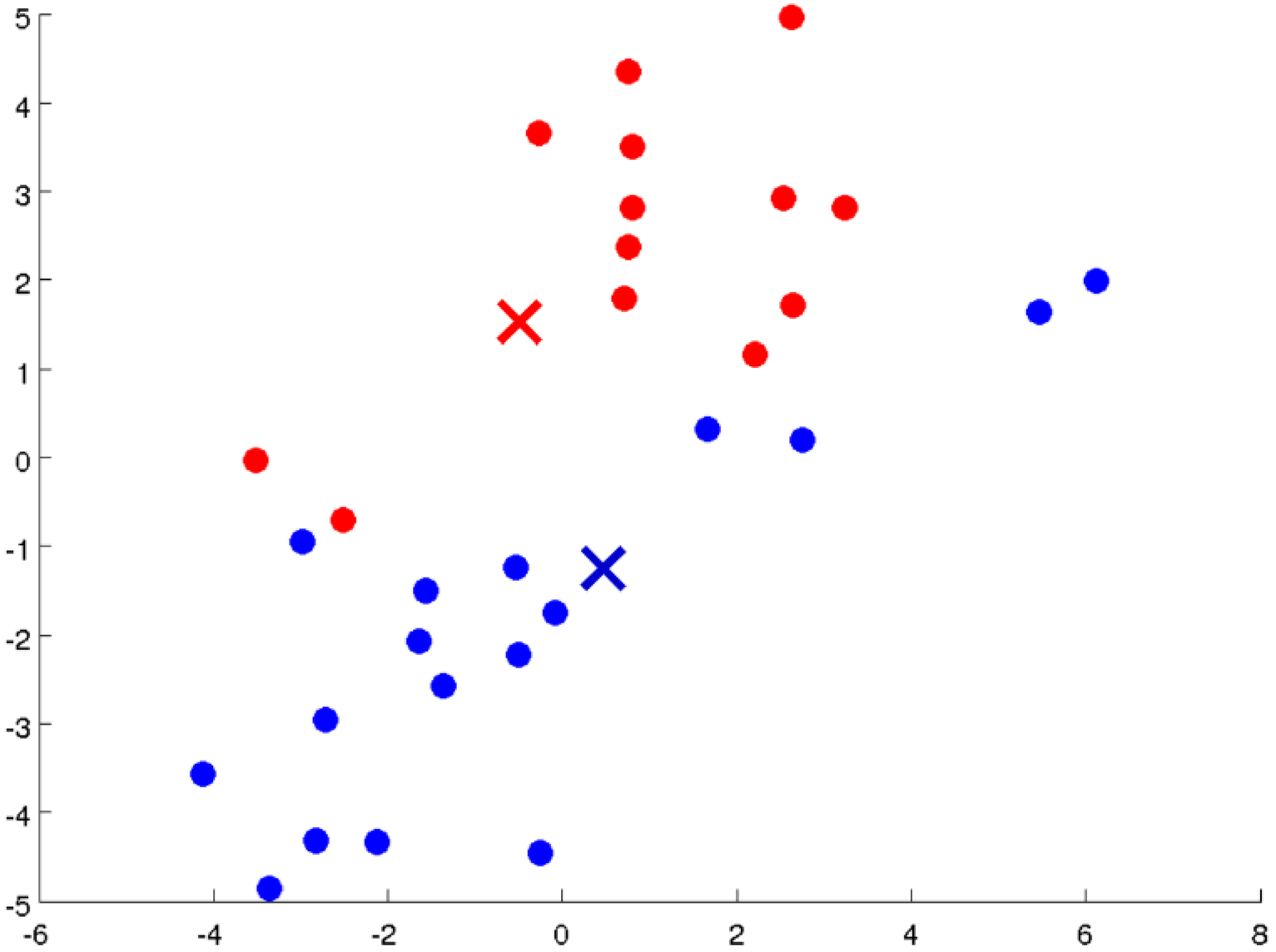
k-means algorithm

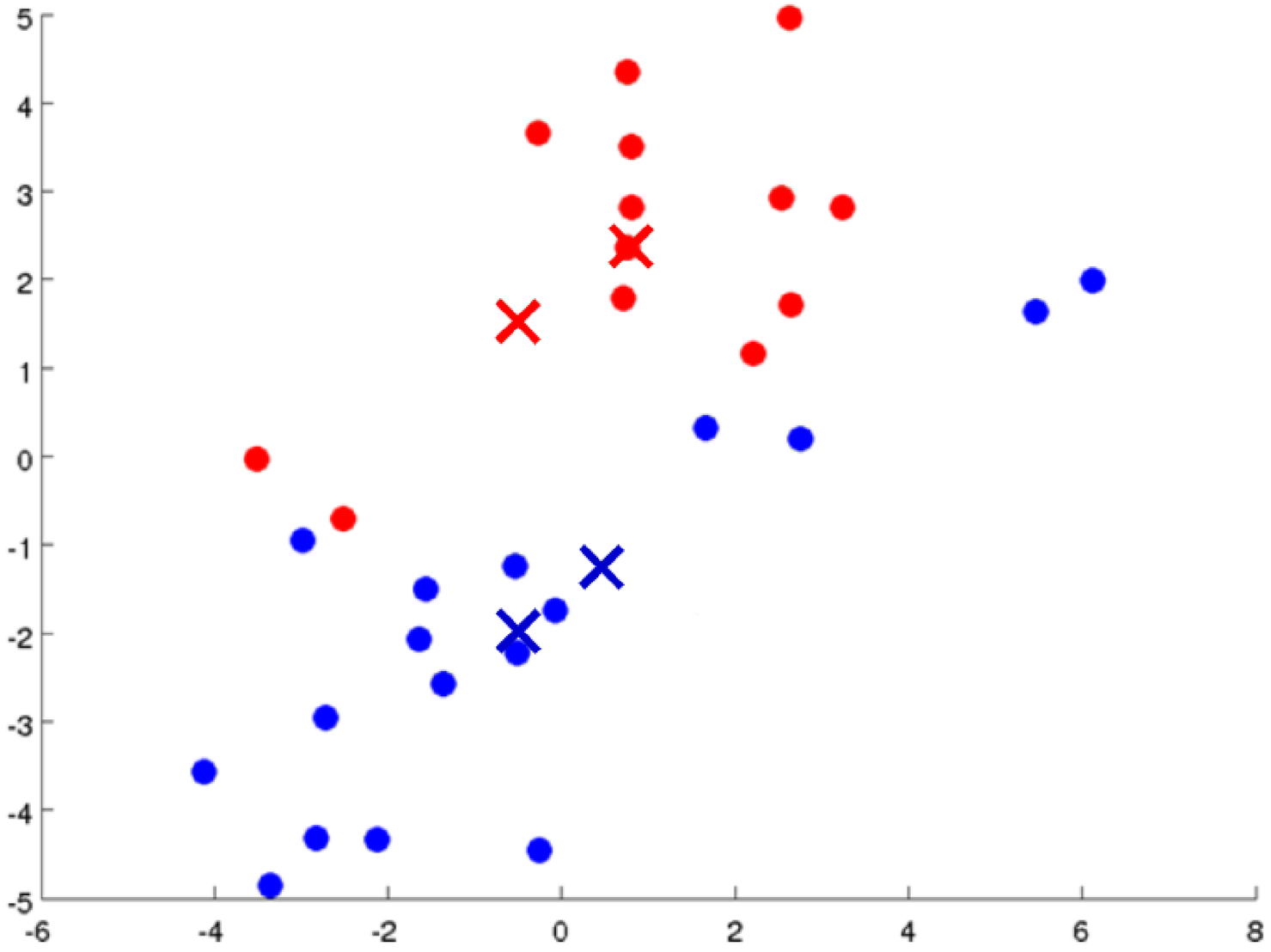

5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8

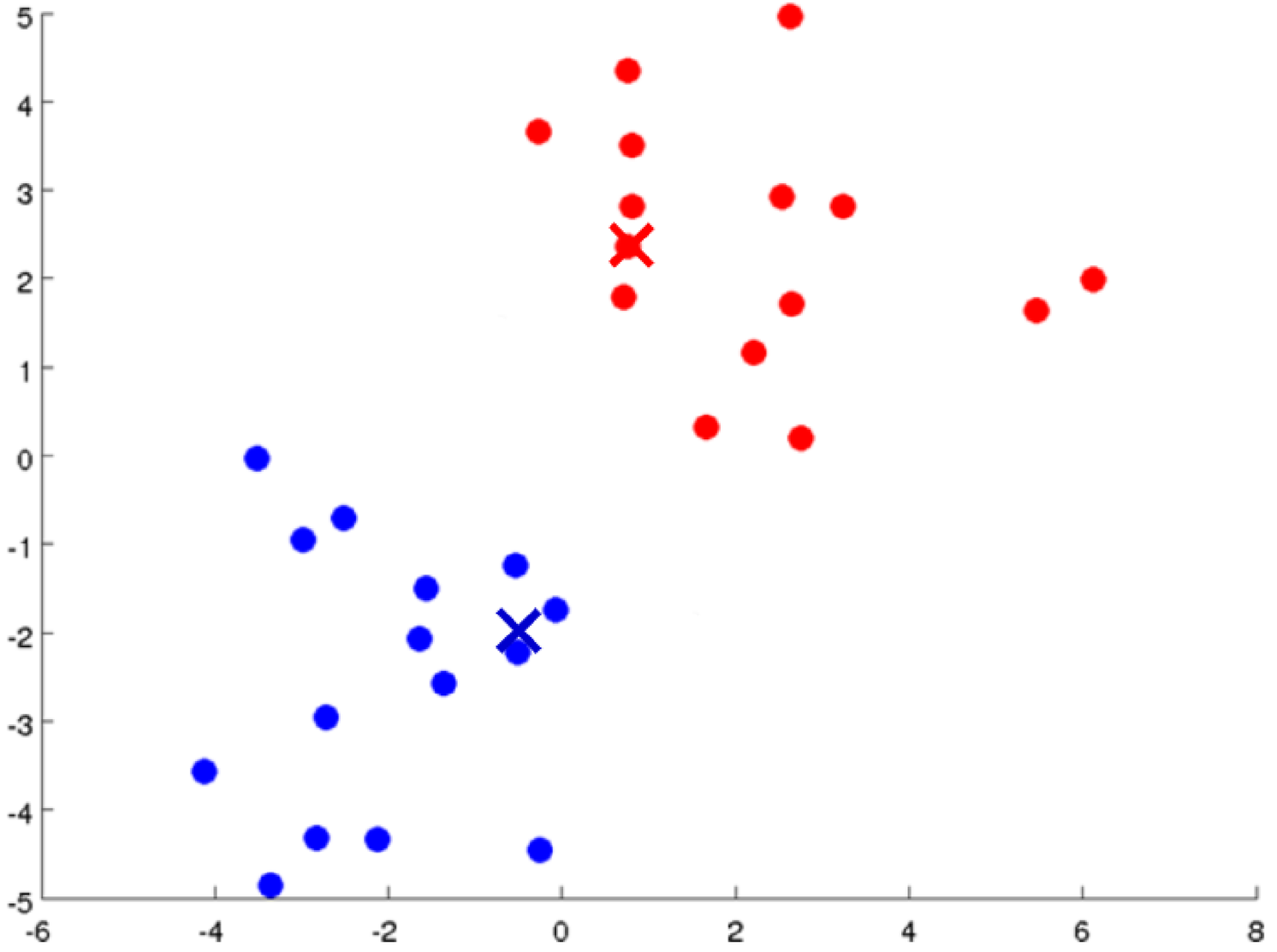

5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8
cluster centroids

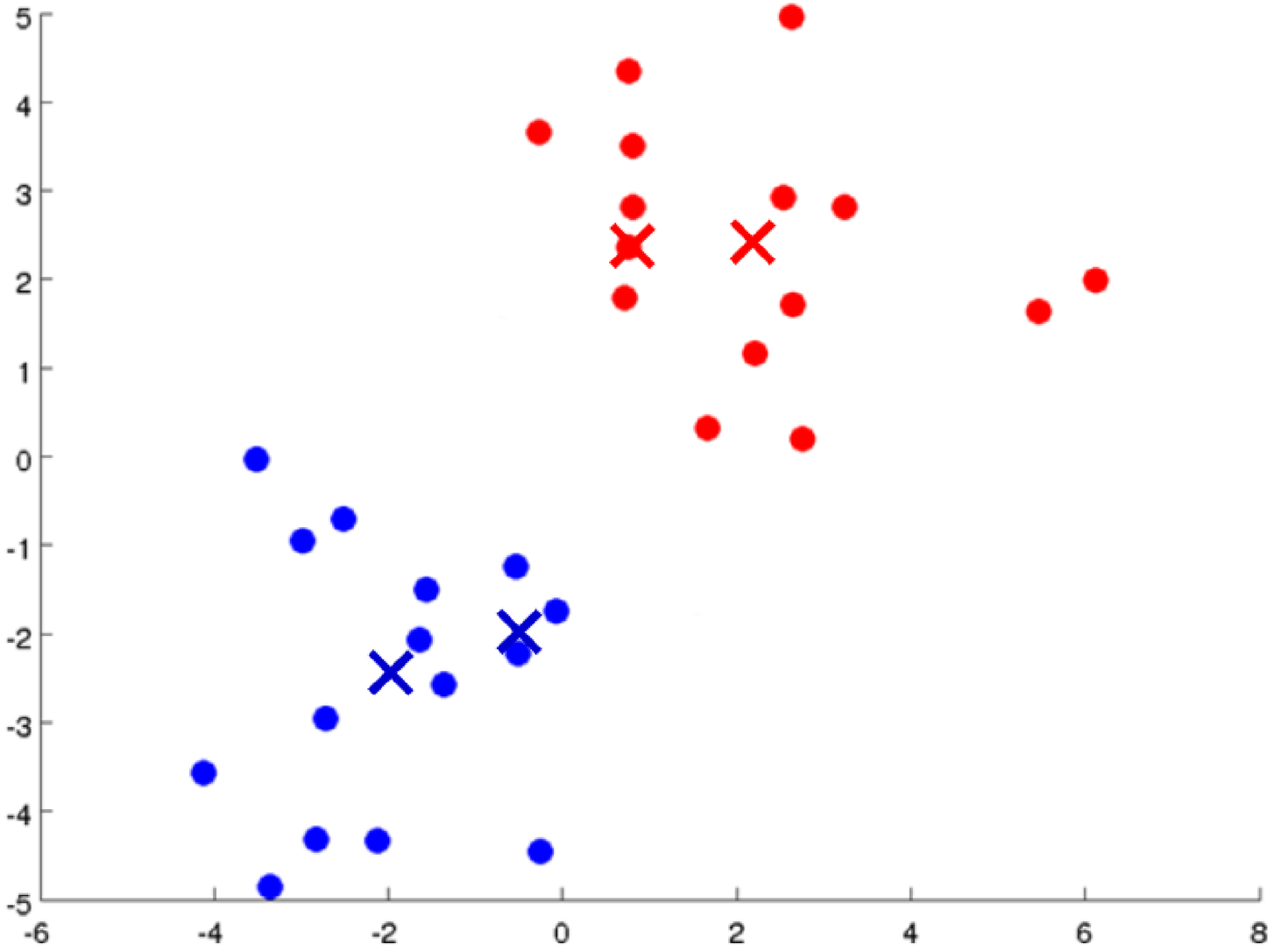

5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8

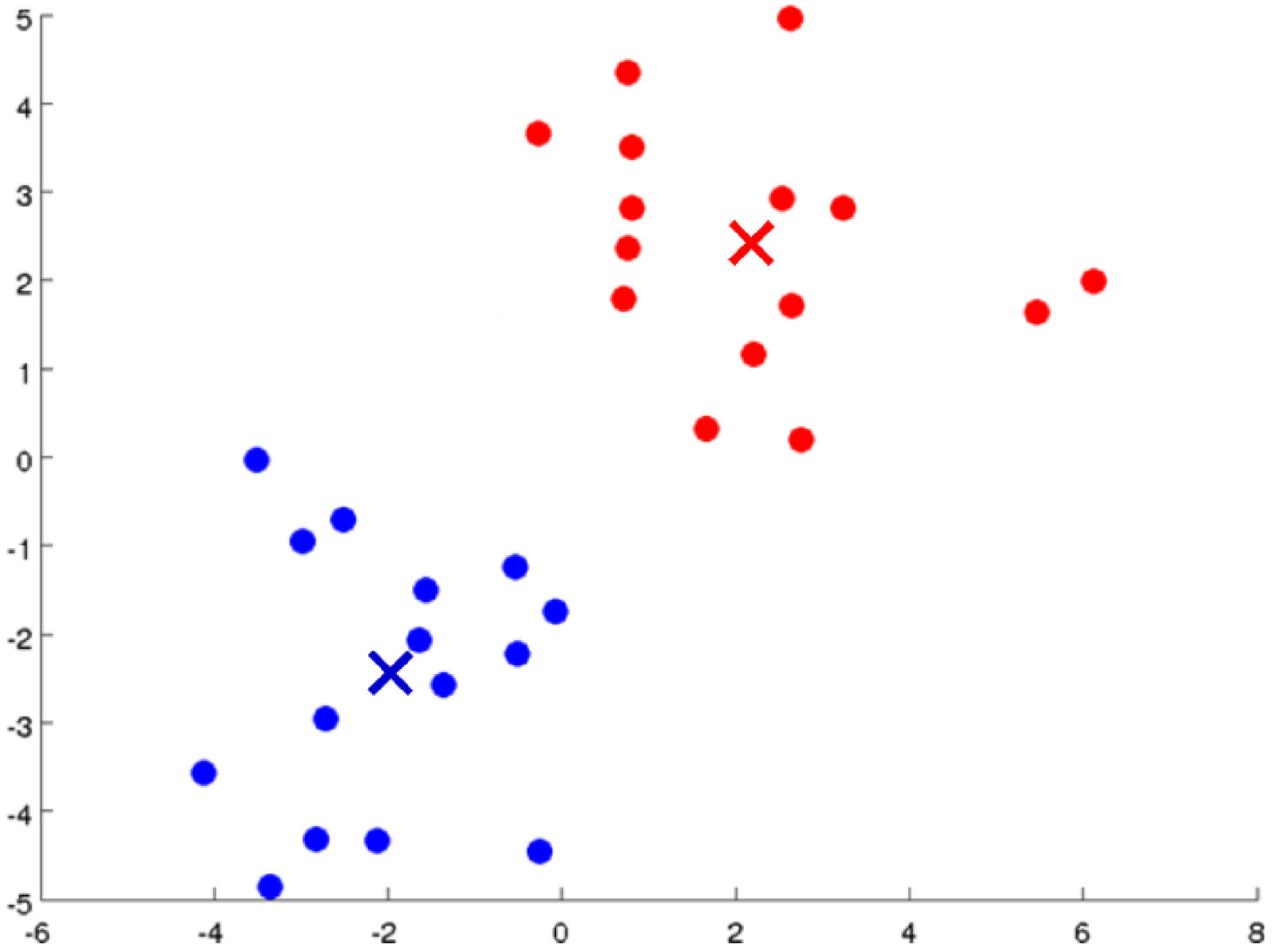

5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8

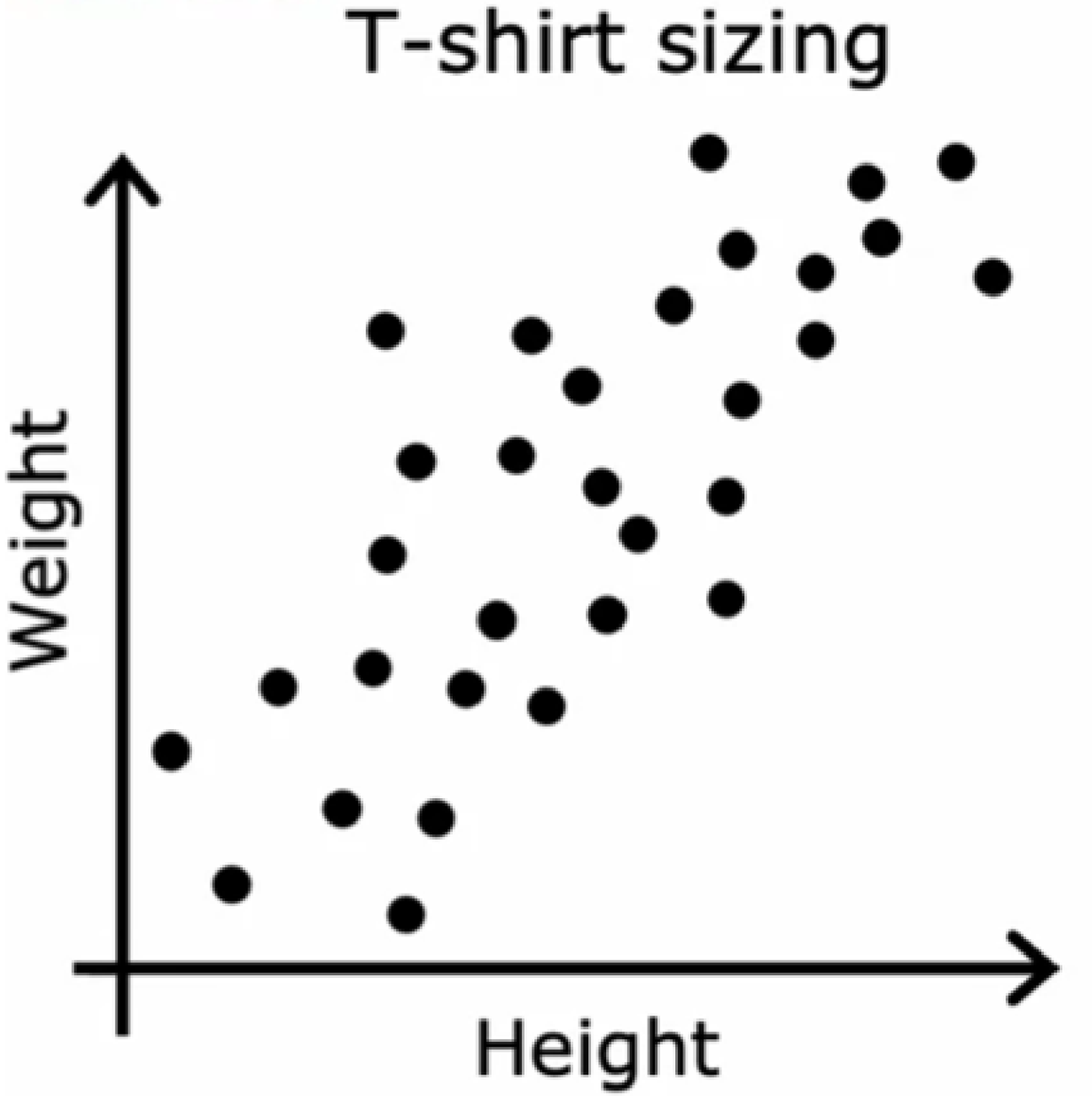

5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


T-shirt sizing
Weight
Height

k-means algorithm-

Step 0: Randomly initialise k cluster centroids.

Repeat {

Step 1: Assign points to cluster centroids

Step 2: Move cluster centroids.

}

K-means optimization objective

$c^{(i)}$ = index of cluster (1,2,...,K) to which example $x^{(i)}$ is currently assigned

μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$)

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned

Optimization objective:

$$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

$$\min_{\substack{c^{(1)},\ldots,c^{(m)},\\ \mu_1,\ldots,\mu_K}} J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$$

K-means algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

for i = 1 to m

$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

for k = 1 to K

μ_k := average (mean) of points assigned to cluster k

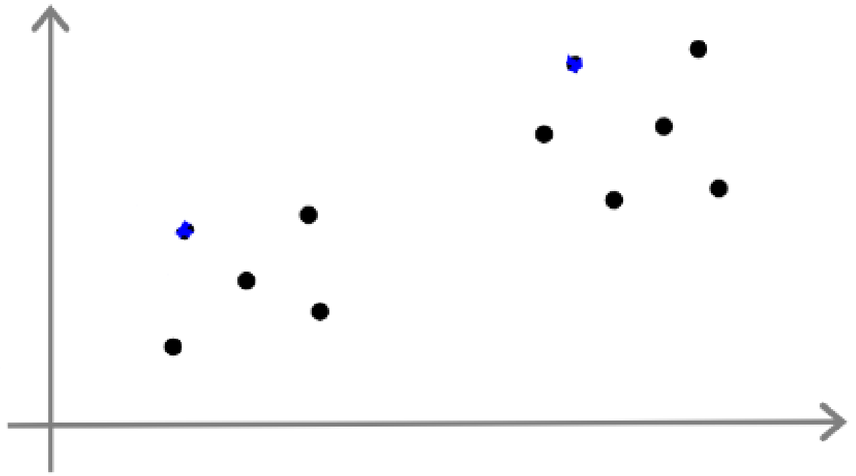
}

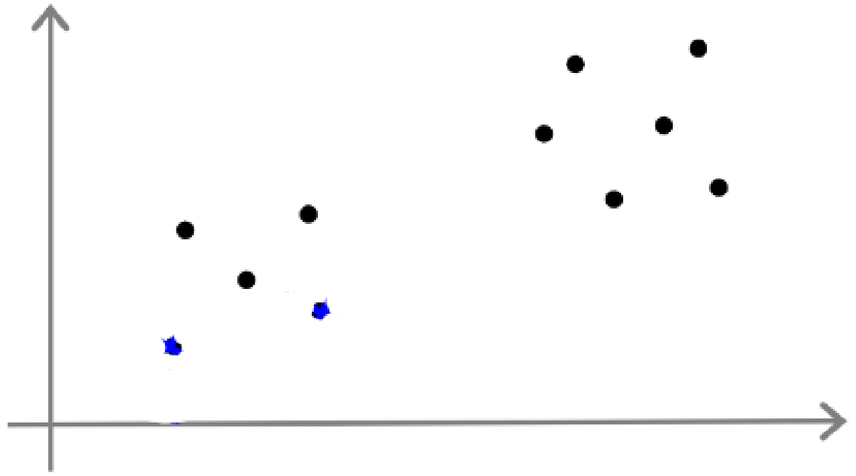
Random initialization

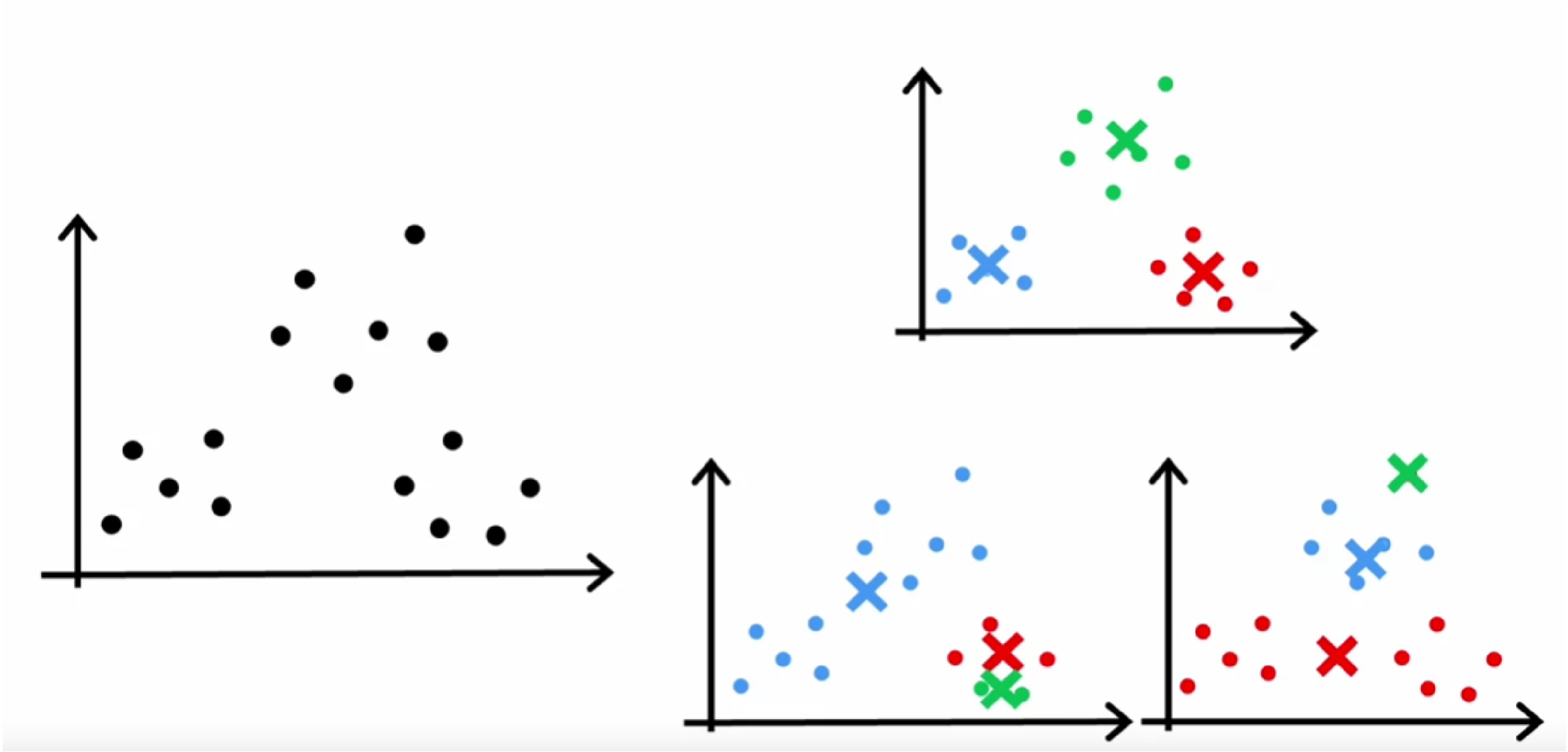
Should have $K < m$

Randomly pick K training examples.

Set $\mu_1, \ldots, \mu_K$ equal to these K examples.







Random initialization

For i = 1 to 100 {

Randomly initialize K-means.

Run K-means. Get $c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K$.

Compute cost function (distortion)

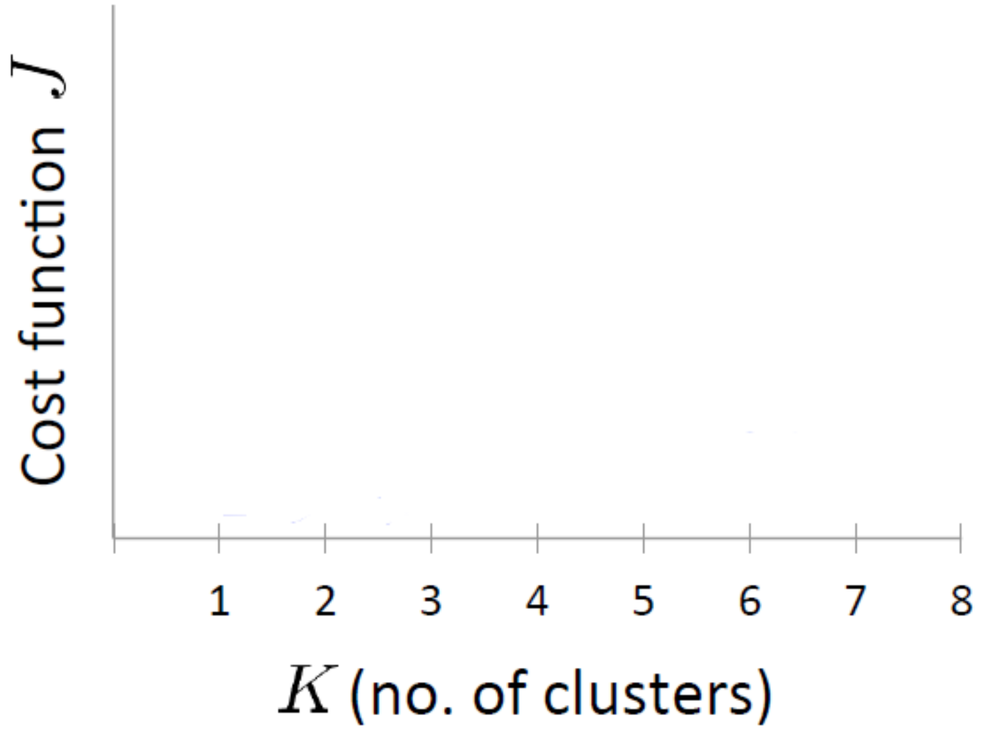
$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

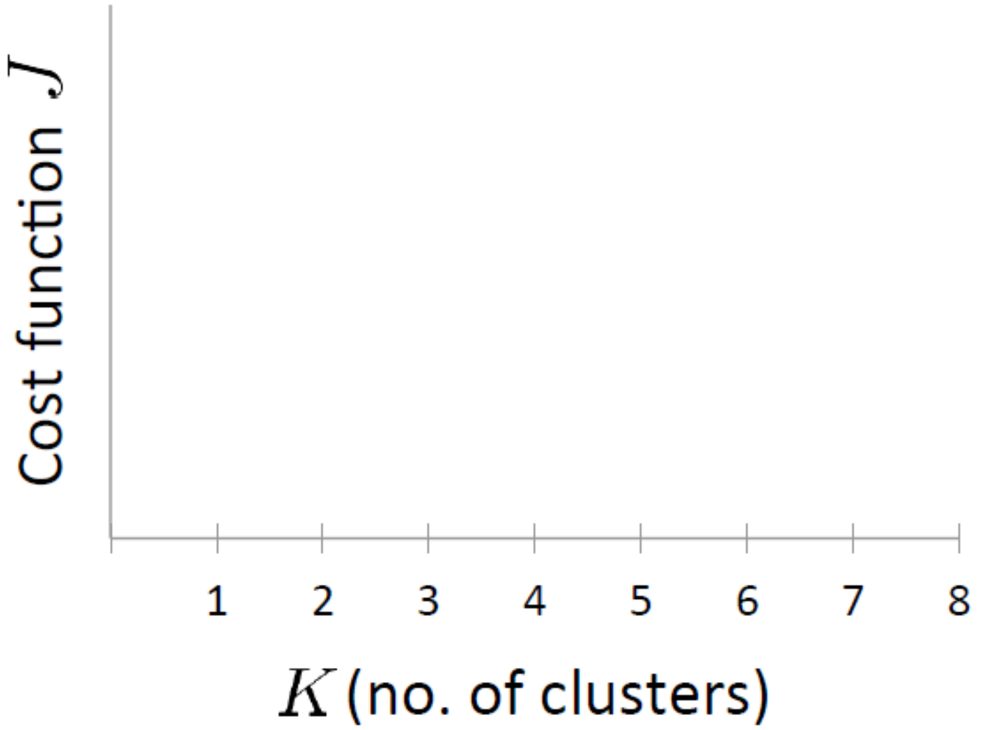
}

Pick clustering that gave lowest cost $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K)$

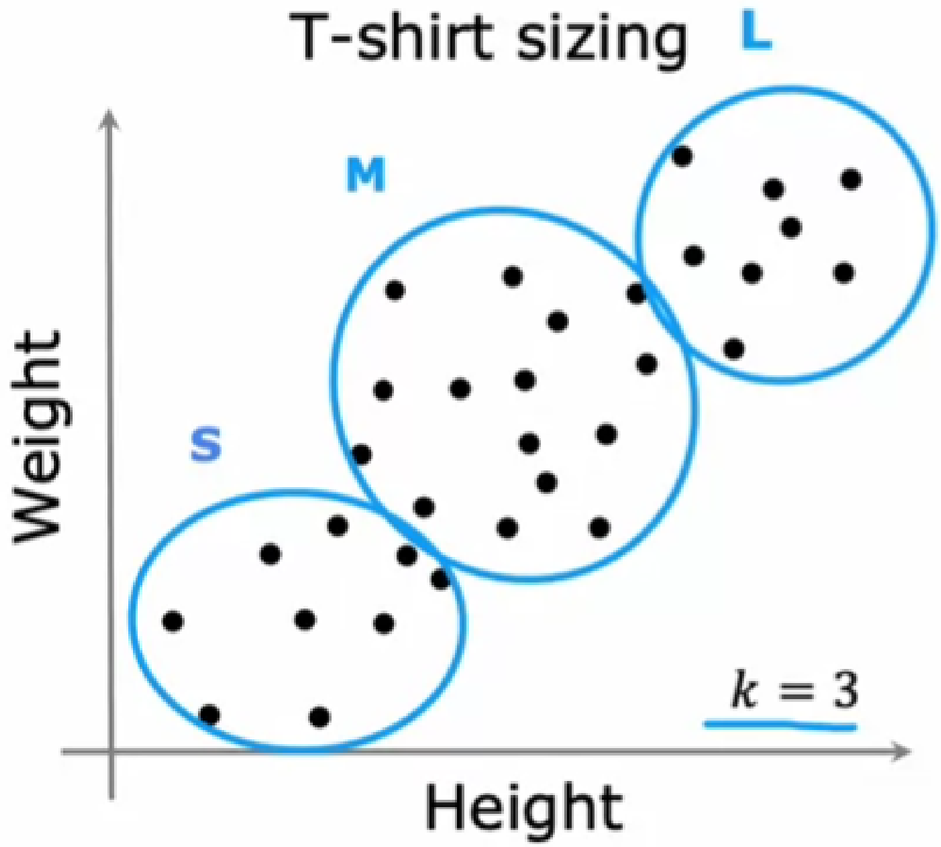
Choosing the value of K:

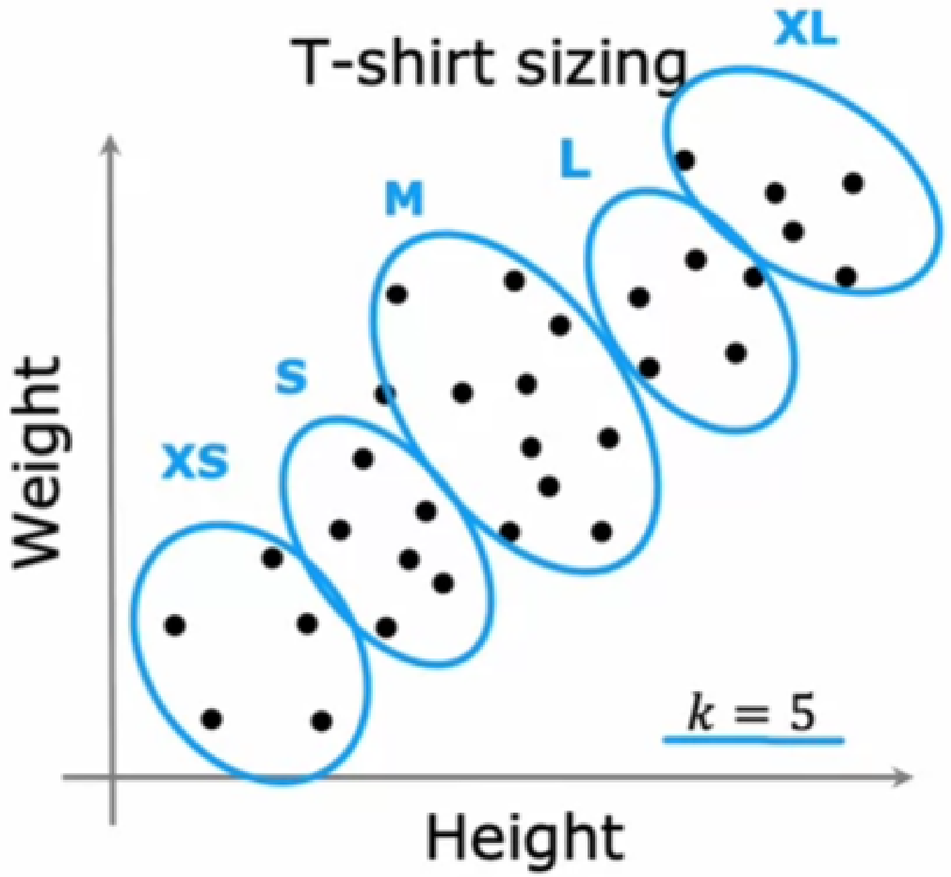
The elbow method-





The value of K might also depend on your specific needs



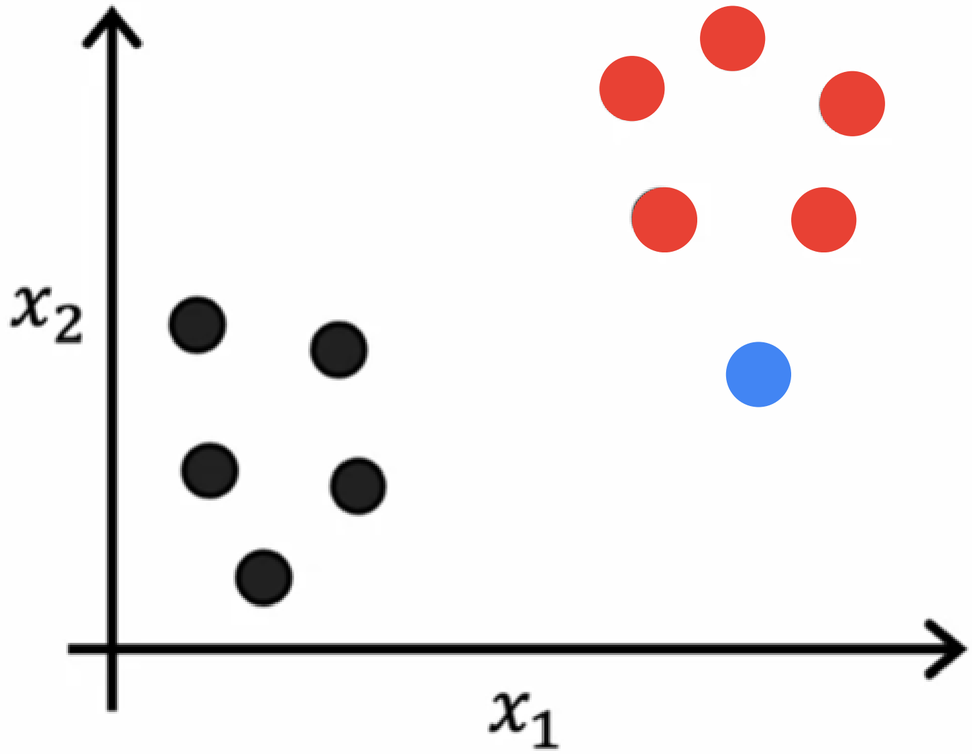


let's code...

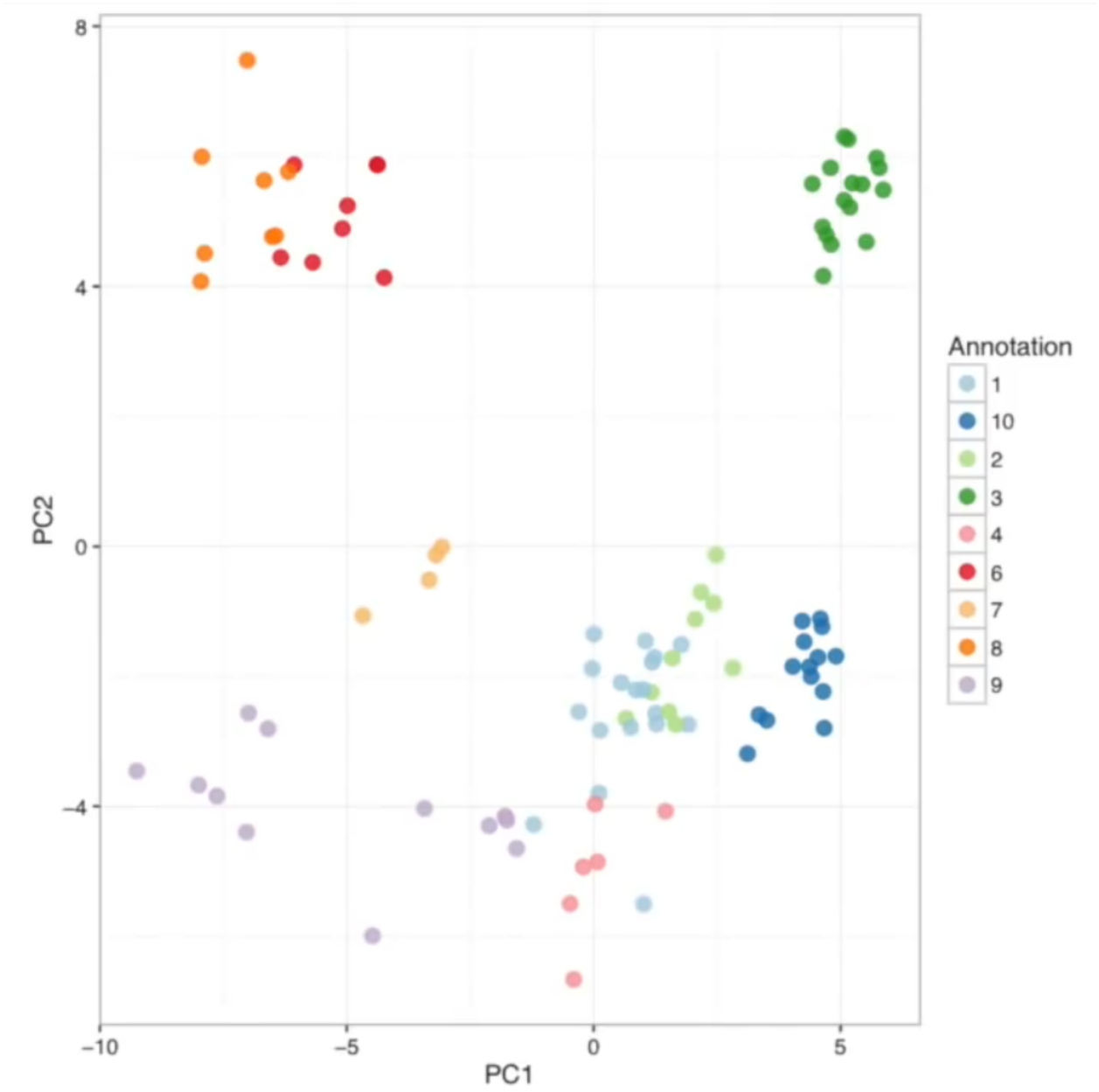
k-nearest neighbours algorithm

k-nearest neighbours

- This algorithm is based on the assumption that data points that are close to each other in space are more likely to belong to the same class.

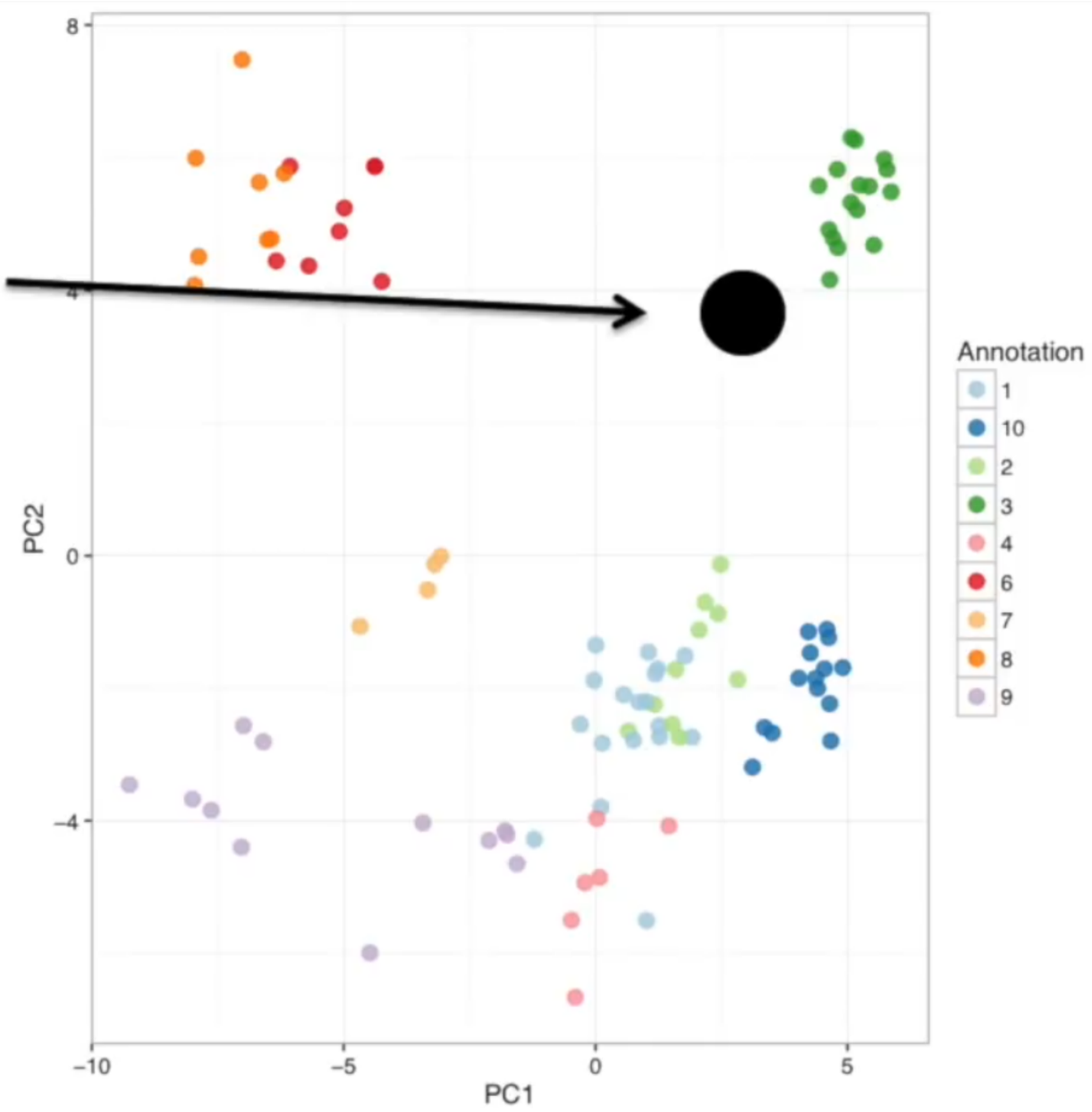


- We start with a dataset with known catagories.



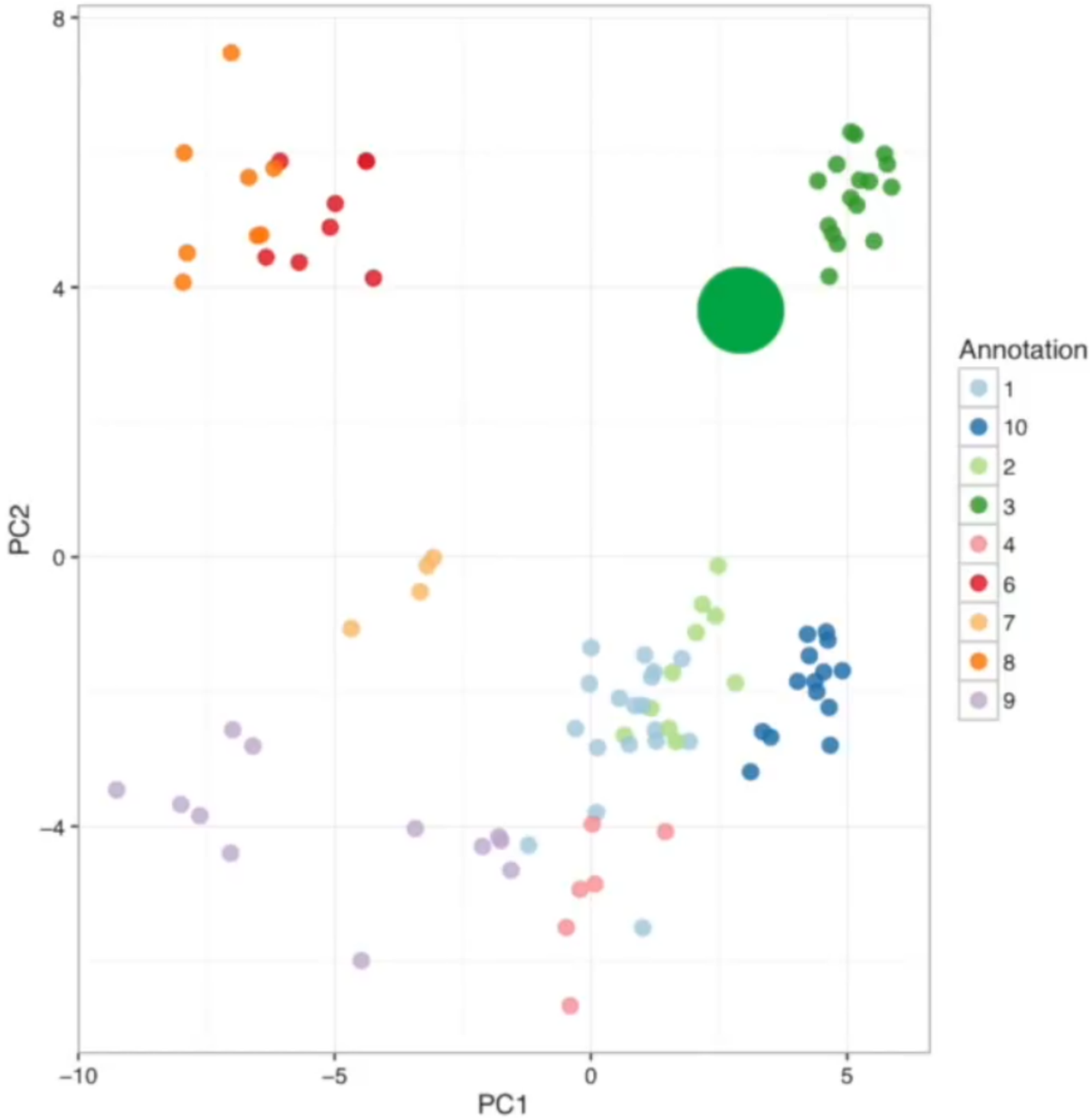
- Now let's say we have a new data point whose category is unknown.
- We classify this data point by looking at its neighbouring points (nearest neighbours).

- If the 'k' in the "k-nearest neighbours" is equal to 1, then we use only the nearest neighbour to define the category.



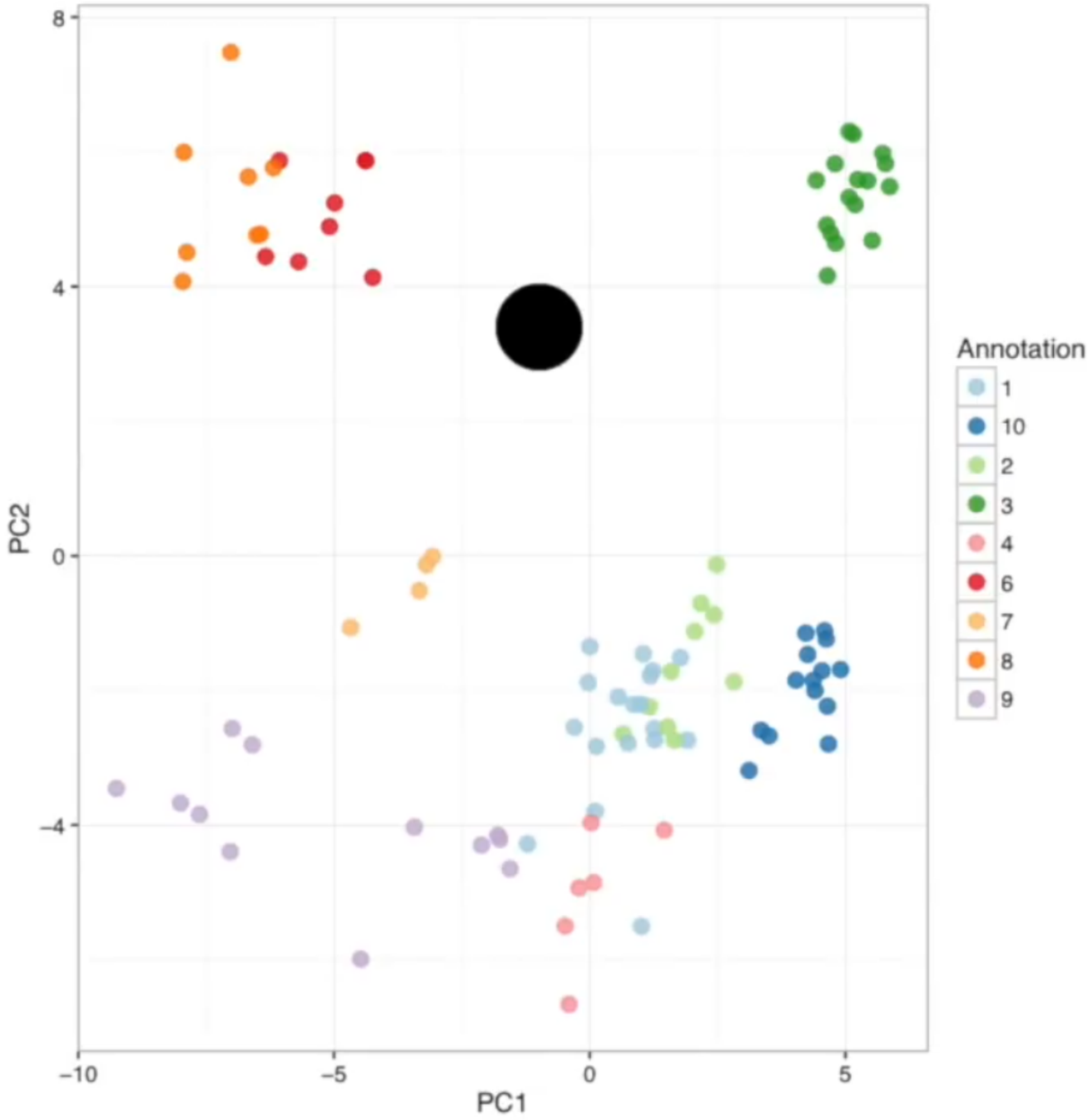
- In this case, the catagory comes out to be red.

- Let's say we had k=11, then we would have to look at the 11 nearest neighbours to define our category.
- In that case, we would look at how many votes each category gets and the category with the maximum number of votes will be the defined catagory.

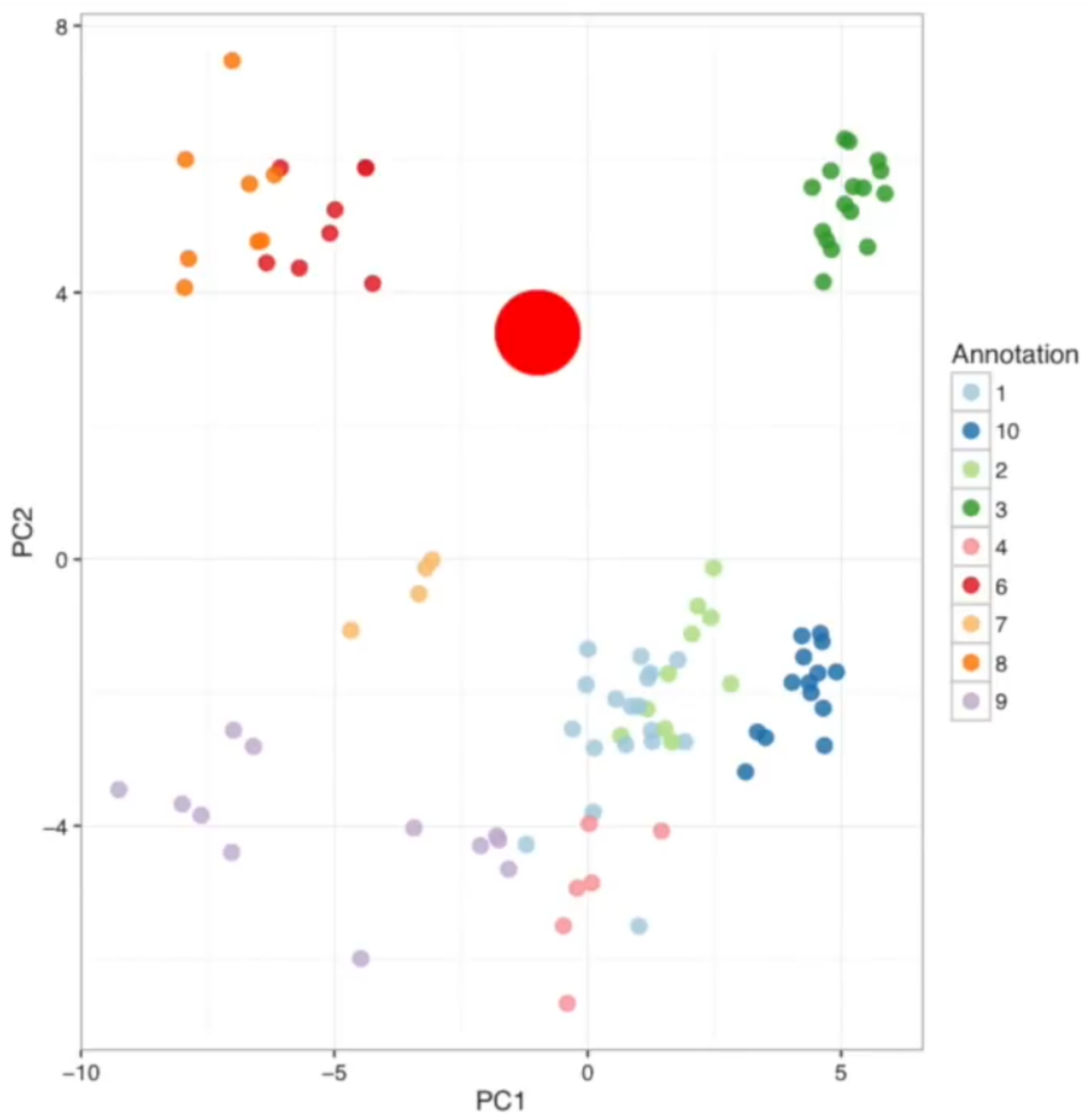


- Let's take a new point which is somewhat between two categories.
- Let us take k=11.

- We look at the 11 nearest neighbours and get the following info-
- 7 red, 3 orange and 1 green neighbours.



- Since red got the most votes, the final assignment is red.



Choosing the value of K:

Somethings to keep in mind-

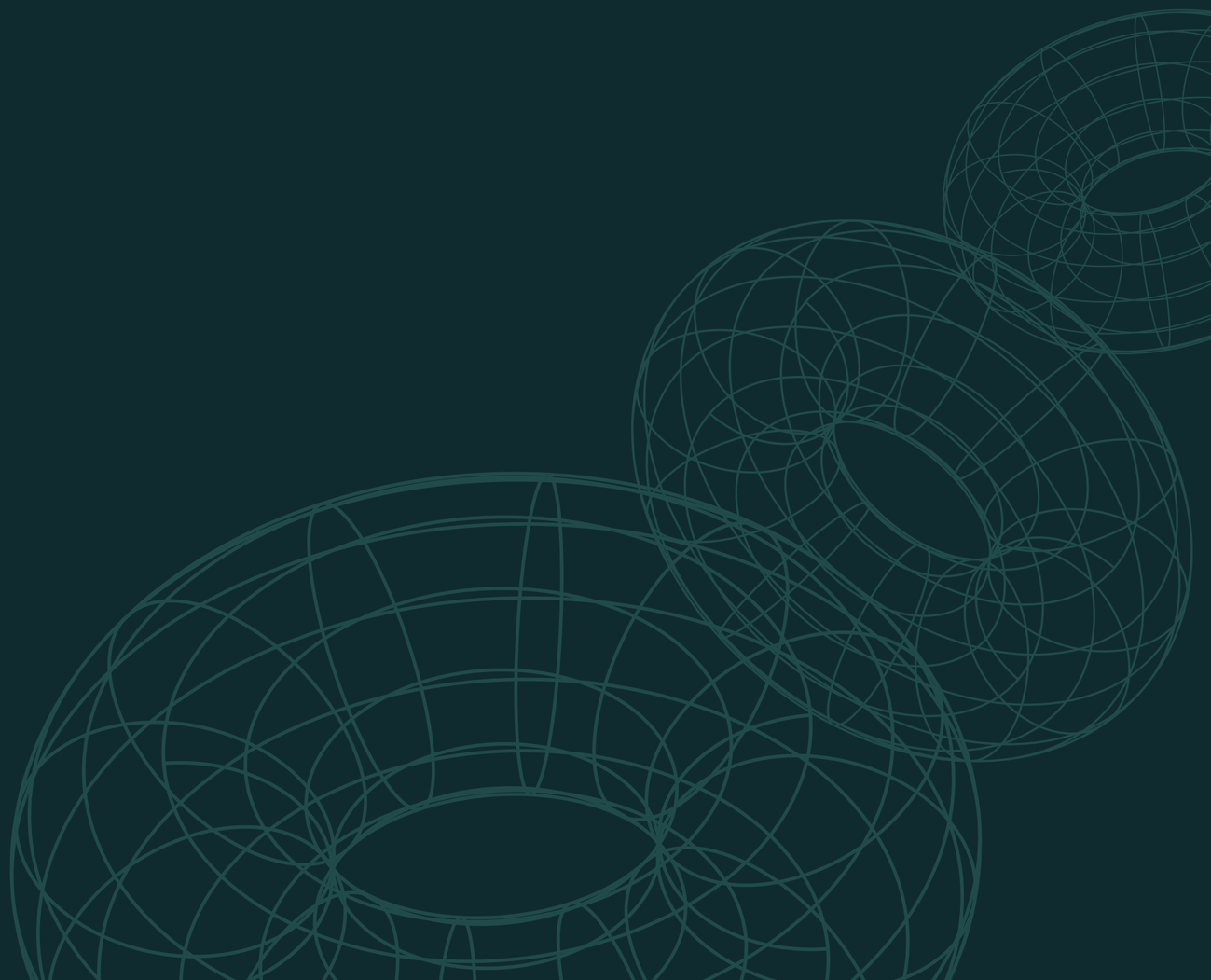
- For small values of K (K= 1 or 2) can be noisy and subject to the effects of outlines.

- Large values for K smooth over things, but you don't want K to be so large that a category with only a few samples in it will always be outvoted by other categories.

Learn with
Google AI | Explore ML

Crowdsource
By Google

Do you have any questions?



Send it to us!

We hope you learned something new.